PySAL
A Python Library of Spatial Analytical Functions

Sergio Rey  Luc Anselin

GEODa CENTER
FOR
GEOSPATIAL ANALYSIS AND COMPUTATION
School of Geographical Sciences and Urban Planning
Arizona State University

Python for Scientific Computing Conference, 2009
Outline

1. Origins
   - GeoDa and STARS
   - Goals and Organization

2. PySAL 1.0
   - Components
   - Illustration

3. Future
   - Organization
   - Invitation
Outline

1. Origins
   - GeoDa and STARS
   - Goals and Organization

2. PySAL 1.0
   - Components
   - Illustration

3. Future
   - Organization
   - Invitation
## PySAL Origins

### CSISS Tools Project

- Center for Spatially Integrated Science
  - NSF infrastructure project 1999-2004
- Tools Project Goals
  - facilitate dissemination of spatial analysis software to social scientists
  - develop a library of spatial data analysis modules
  - develop prototypes implementing state of the art methods
  - initiate and nurture a community of open source developers
Beyond CSISS

PySAL Outgrowth of CSISS Tools

- CSISS Project Closed
- Continued Spatial Tools Development
  - New funding sources
  - NIH/NCI, NSF/HSD
- PySAL Collaborative Effort
  - PySAL: a Python library for spatial analysis
  - UIUC Group: GeoDa, PySpace
  - SDSU Group: STARS
## PySAL Team

### Committers
- Luc Anselin (ASU)
- Serge Rey (ASU)
- Charles Schmidt (ASU)
- Andrew Winslow (ASU, Tufts)

### Contributors
- Daniel Arribas-Bel (University of Zaragoza)
- David Folch (ASU)
- Xinyue Ye (SDSU-UCSB)
Outline

1 Origins
   - GeoDa and STARS
   - Goals and Organization

2 PySAL 1.0
   - Components
   - Illustration

3 Future
   - Organization
   - Invitation
GeoDa Center for Geospatial Analysis and Computation

Activities

- **Develop** state-of-the-art methods for geospatial analysis, geovisualization, geosimulation, and spatial process modeling,
- **Implement** them through software tools
- **Apply** them to policy-relevant research in the social and environmental sciences
- **Disseminate** them through training and support to a growing worldwide geospatial community.
PySAL Philosophy

Guiding Principles

- Scientific glue
- Code as text
PySAL Objectives

Leverage Existing Tools Development
- GeoDa/PySpace
- STARS

Develop Core Library
- spatial data analytical functions
- enhanced specialization, modularization
- fill a void in Python libraries

Flexible Delivery Mechanism
- GUI, ArcGIS interface
- spatial analytical web services
Outline

1 Origins
   - GeoDa and STARS
   - Goals and Organization

2 PySAL 1.0
   - Components
   - Illustration

3 Future
   - Organization
   - Invitation
Outline

1. Origins
   - GeoDa and STARS
   - Goals and Organization

2. PySAL 1.0
   - Components
   - Illustration

3. Future
   - Organization
   - Invitation
Using PySAL: regular Python module

```python
serge@highfalls:~$ python
Python 2.5 (r25:51918, Sep 19 2006, 08:49:13)
[GCC 4.0.1 (Apple Computer, Inc. build 5341)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pysal
>>> pysal.FileIO.check()
PySAL File I/O understands the following file extensions:
Ext: '.shp', Modes: ['r', 'w']
Ext: '.dbf', Modes: ['r']
Ext: '.gwt', Modes: ['r']
Ext: '.gal', Modes: ['r']
Ext: '.csv', Modes: ['r']
Ext: '.wkt', Modes: ['r']
>>> 
```
Spatial Weights Component

- MST
- Voronoi
- Hulls
- Computational Geometry
- GAL
- GWT
- MAT
- Estimation
- Testing
- Diagnostics
- Simulation
- Spatial Econometrics
- ESDA
- Spatial Dynamics
- Smoothing
- LISA
- Spatial ϑ
- Spatial τ
- Spatial Markov

Origins
PySAL 1.0
Future
Illustration
Rey and Anselin
PySAL
Spatial Weights from Shapefiles
Spatial Weights from Shapefiles

In [1]: fname="US_Tract_2000.shp"
In [2]: shpFile=shpIO.shpFile(fname)
In [3]: c=rook(shpFile)
In [4]: c.max_neighbors
Out[4]:
30
In [4]: list(c.cardinalities).index(c.max_neighbors)
Out[4]:
24089
Spatial Weights: Cardinalities
Spatial Weights Construction

```
In [1]: from pysal.weights import weights
In [2]: w=weights.lat2gal(3,3)
In [3]: w.weights
Out[3]:
{0: [1, 1],
  1: [1, 1, 1],
  2: [1, 1],
  3: [1, 1, 1],
  4: [1, 1, 1, 1],
  5: [1, 1, 1],
  6: [1, 1],
  7: [1, 1, 1],
  8: [1, 1]}
In [4]: w.neighbors
Out[4]:
{0: [3, 1],
  1: [0, 4, 2],
  2: [1, 5],
  3: [0, 6, 4],
  4: [1, 3, 7, 5],
  5: [2, 4, 8],
  6: [3, 7],
  7: [4, 6, 8],
  8: [5, 7]}
```
Spatial Weights Transformations

In [5]: w.transform='w'
In [6]: w.weights
Out[6]:
{0: [0.5, 0.5],
  1: [0.3333333333333333, 0.3333333333333333, 0.3333333333333333],
  2: [0.5, 0.5],
  3: [0.3333333333333333, 0.3333333333333333, 0.3333333333333333],
  4: [0.25, 0.25, 0.25, 0.25],
  5: [0.3333333333333333, 0.3333333333333333, 0.3333333333333333],
  6: [0.5, 0.5],
  7: [0.3333333333333333, 0.3333333333333333, 0.3333333333333333],
  8: [0.5, 0.5],
}
In [7]: w.transform='v'
In [8]: w.weights
Out[8]:
{0: [0.43633431266944905, 0.43633431266944905],
  1: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  2: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  3: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  4: [0.30853495135293874, 0.30853495135293874, 0.30853495135293874],
  5: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  6: [0.43633431266944905, 0.43633431266944905],
  7: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  8: [0.43633431266944905, 0.43633431266944905]}
Spatial Weights Transformations: Back to Binary

In [7]: w.transform='v'
In [8]: w.weights
Out[8]:
{0: [0.43633431266944905, 0.43633431266944905],
  1: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  2: [0.43633431266944905, 0.43633431266944905],
  3: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  4: [0.30853495135293874, 0.30853495135293874, 0.30853495135293874, 0.30853495135293874],
  5: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  6: [0.43633431266944905, 0.43633431266944905],
  7: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  8: [0.30853495135293874, 0.30853495135293874, 0.30853495135293874],
  0.30853495135293874, 0.30853495135293874, 0.30853495135293874],
  12: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  13: [0.43633431266944905, 0.43633431266944905],
  14: [0.3562654744360545, 0.3562654744360545, 0.3562654744360545, 0.3562654744360545],
  15: [0.43633431266944905, 0.43633431266944905, 0.43633431266944905]}
In [9]: w.transform='b'
In [10]: w.weights
Out[10]:
{0: [1, 1],
  1: [1, 1, 1],
  2: [1, 1],
  3: [1, 1, 1],
  4: [1, 1, 1, 1],
  5: [1, 1, 1],
  6: [1, 1],
  7: [1, 1, 1],
  8: [1, 1]}
In [11]:
Spatial Weights: transformation as a property

1  In [18]: w.s1
2  Out[18]: 48.0
3
4  In [19]: w.transform='v'
5
6  In [20]: w.s1
7  Out[20]: 6.7935538046744153
8
9  In [21]: w.transform='b'
10
11 In [22]: w.s1
12 Out[22]: 48.0
13
Spatial Weights: transformation as a property

\[ s_1 = \frac{1}{2} \sum_i \sum_j (w_{i,j} + w_{j,i})^2 \]
Spatial Weights Characteristics

In [14]: w.s0,w.s1,w.s2,w.pct_nonzero
Out[14]: (24.0, 48.0, 272.0, 0.29629629629629628)

In [15]: w.

w.__class__  w.__reduce__  w.full  w.pct_nonzero
w.__delattr__ w.__reduce_ex__ w.get_transform w.s0
w.__dict__  w.__repr__  w.lag  w.s1
w.__doc__  w.__setattr__  w.max_neighbors  w.s2
w.__getattribute__  w.__str__  w.mean_neighbors  w.sd
w.__hash__  w.__weakref__  w.min_neighbors  w.set_transform
w.__init__  w._transform  w.n  w.transform
w.__module__  w.cardinalities  w.neighbors  w.transformations
w.__new__  w.characteristics  w.nonzero  w.weights
w.weights
Spatial Weights Methods: full

```python
1 In [15]: wf=w.full()
2 3 In [16]: wf
4 Out[16]:
5 array([[ 0., 1., 0., 1., 0., 0., 0., 0., 0.],
6   [ 1., 0., 1., 0., 1., 0., 0., 0., 0.],
7   [ 0., 1., 0., 0., 0., 1., 0., 0., 0.],
8   [ 1., 0., 0., 0., 1., 0., 1., 0., 0.],
9   [ 0., 1., 0., 1., 0., 0., 1., 0., 0.],
10  [ 0., 0., 1., 0., 1., 0., 0., 0., 1.],
11  [ 0., 0., 0., 1., 0., 0., 0., 1., 0.],
12  [ 0., 0., 0., 0., 1., 0., 1., 0., 1.],
13  [ 0., 0., 0., 0., 0., 1., 0., 1., 0.]]
14 In [17]:
```
Spatial Weights Methods: lag

```python
In [35]: import numpy as num
In [36]: y=num.arange(9)
In [37]: w.transform='o'
In [38]: ylb=w.lag(y)
In [39]: w.transform='w'
In [40]: ylw=w.lag(y)
In [41]: ylb
Out[41]: array([ 4., 6., 6., 10., 16., 14., 10., 18., 12.])
In [42]: ylw
Out[42]: array([ 2. , 2. , 3. , 3.33333333, 4. , 4.66666667, 5. , 6. , 6. ])
```
def higher_order(self, k=3):
    """Return contiguity weights object of order k (default k=3)
    Implements the algorithm in Anselin and Smirnov (1996)
    Example Usage:

    >>> w5 = lat2gal()
    >>> w5_shimbel = w5.shimbel()  
    >>> w5_shimbel[0][24]
    8
    >>> w5_shimbel[0][0:4]
    [-1, 1, 2, 3]
    >>> w5_8th_order = w5.higher_order(8)
    >>> w5_8th_order.neighbors[0]
    [24]
    >>>
    """
ESDA: Global Moran’s I

```
1 In [1]: from pysal.core._FileIO.file_io import load, read_GAL
2 In [2]: w = read_GAL("../data/stl.GAL")
3 In [3]: variables = load("../data/stl_hom.txt", file_type="geoda_txt")
4 In [4]: hr8893 = variables.get_variable('HR8893')
5 In [5]: from pysal.esda.moran import Moran
6 In [6]: mi = Moran(hr8893, w)
7 In [7]: mi.
8    mi.EI    mi.__class__   mi.p_sim   mi.y
9    mi.EI_sim mi.__doc__   mi.p_z_sim mi.z
10   mi.I      mi.__init__  mi.permutations mi.z2ss
11   mi.VI_norm mi.__module__ mi.seI_norm mi.z_norm
12   mi.VI_rand mi.inum     mi.seI_rand mi.z_rand
13   mi.VI_sim  mi.n        mi.seI_sim  mi.z_sim
14  mi._Moran__calc  mi.p_norm mi.sim
15 mi._Moran__moments  mi.p_rand mi.w
16 In [7]: mi.I
17 Out[7]: 0.2436582621771661
18 In [8]: mi.p_norm, mi.p_rand, mi.p_sim
19 Out[8]:
20    (array(0.00052730423329256173),
21     array(0.0005298110894588466),
22     0.02100000000000001)
```
ESDA: Global Moran’s I, with transformation option

In [9]: w.transform
Out[9]: 'W'

In [10]: mib=Moran(hr8893, w, transformation='b')

In [11]: mib.I
Out[11]: 0.20344119287936516

In [12]: mib.p_norm, mib.p_rand, mib.p_sim
Out[12]:
(array(0.0025436023351127763),
array(0.0025824890368620992),
0.021000000000000001)
ESDA: Local Moran’s I

```
In [13]: from pysal.esda.moran import Moran_Local
In [14]: ml=Moran_Local(hr8893,w)
In [15]: ml.
ml.EI_sim ml.__init__ ml.seI_sim
ml.Is ml.__module__ ml.sim
ml.VI_sim ml.den ml.w
ml._Moran_Local__calc ml.p_sim ml.y
ml._Moran_Local__quads ml.p_z_sim ml.z
ml.__class__ ml.permutations ml.z_sim
ml.__doc__ ml.q
In [16]: ml.Is
Out[16]:
```

```-snip->
Rey and Anselin
```
ESDA: Local Moran’s I (cont.)

In [18]: ml.q
Out[18]:
array([[2, 3, 3, 3, 4, 4, 3, 2, 3, 3, 3, 3, 3, 4, 4, 3,
   3, 3, 3, 3, 3, 4, 1, 4, 2, 3, 1, 3, 3, 4, 3, 2, 1,
   3, 3, 3, 3, 3, 2, 3, 1, 4, 3, 3, 3, 2, 3, 4, 3, 4, 2, 3, 1,
   2, 2, 3, 3, 1, 4, 3, 2, 1]])

In [19]: ml.p_sim
Out[19]:
array([[ 0.2 , 0.081, 0.402, 0.291, 0.377, 0.051, 0.278, 0.245,
   0.047, 0.053, 0.249, 0.545, 0.467, 0.655, 0.424, 0.495,
   0.501, 0.583, 0.601, 0.192, 0.124, 0.036, 0.348, 0.597,
   0.276, 0.661, 0.194, 0.325, 0.42 , 0.584, 0.498, 0.531,
   0.005, 0.551, 0.256, 0.017, 0.004, 0.003, 0.065, 0.045,
   0.085, 0.427, 0.554, 0.294, 0.248, 0.022, 0.031, 0.049,
   0.06 , 0.117, 0.311, 0.322, 0.099, 0.498, 0.052, 0.295,
   0.11 , 0.653, 0.677, 0.655, 0.3 , 0.629, 0.363, 0.501,
   0.368, 0.274, 0.27 , 0.445, 0.177, 0.177, 0.245, 0.594,
   0.545, 0.147, 0.612, 0.562, 0.077, 0.12 ]])
In [20]: variables.var_names
Out[20]: ['FIPSNO', 'HR8488', 'HR8893', 'HC8488']
In [21]: hc8488=variables.get_variable('HC8488')
In [22]: from pysal.esda.moran import Moran_BV
In [23]: mbv=Moran_BV(hr8893,hc8488,w)
In [24]: mbv.
    mbv.EI_sim  mbv.__module__  mbv.y1
    mbv.I  mbv.inum  mbv.y2
    mbv.VI_sim  mbv.p_sim  mbv.z1
    mbv._Moran_BV__calc  mbv.p_z_sim  mbv.z12ss
    mbv.__class__  mbv.seI_sim  mbv.z_sim
    mbv.__doc__  mbv.sim
    mbv.__init__  mbv.w
In [24]: mbv.I
Out[24]: 0.28288633291435145
In [25]: mbv.p_sim
Out[25]: 0.01
In [26]: mbv2=Moran_BV(hc8488,hr8893,w)
In [27]: mbv2.I
Out[27]: 0.28087192977483516
In [28]: mbv2.p_sim
Out[28]: 0.01
ESDA: Bivariate Moran’s I Matrix

```
In [29]: v3=variables.var_names[1:]
In [30]: v3
Out[30]: ['HR8488', 'HR8893', 'HC8488']
In [31]: ys=[variables.get_variable(v) for v in v3]
In [32]: from pysal.esda.moran import Moran_BV_matrix as bvm
In [33]: matres=bvm(ys,w)
In [34]: matres.keys()
Out[34]: [(0, 1), (1, 2), (2, 1), (2, 0), (1, 0), (0, 2)]
In [35]: for key in matres.keys():
   ....:     i,j=key
   ....:     print v3[i],v3[j],matres[key].p_sim
   ....:
HR8488 HR8893 0.017
HR8893 HC8488 0.01
HC8488 HR8488 0.008
HC8488 HR8488 0.022
HR8893 HR8488 0.024
HR8488 HC8488 0.008
```
Fortune’s Plane Sweep Algorithm

Algorithm

- $O(n \log n)$ time
- $O(n)$ space
- beautiful algorithm
Fortune’s Plane Sweep Algorithm

Algorithm

- $O(n \log n)$ time
- $O(n)$ space
- beautiful algorithm
Fortune’s Plane Sweep Algorithm

Algorithm
- $O(n \log n)$ time
- $O(n)$ space
- Beautiful algorithm

PySAL Implementation
- Pure Python
- Pedagogy
- Decent speed
In [7]: run vweights
generating point set: 0
1000 1
Points to Voronoi Weights

In [7]: run vweights

generating point set: 0

1000 1

Voronoi time to solve: 2.2035
Outline

1 Origins
   - GeoDa and STARS
   - Goals and Organization

2 PySAL 1.0
   - Components
   - Illustration

3 Future
   - Organization
   - Invitation
PySAL: Open Source Python Library for Spatial Analytical Functions

PySAL is a collaborative effort between Luc Anselin and Sergio Rey to develop a cross-platform library of spatial analysis functions (see figure) written in Python®. It is a work-in-progress that combines the development efforts on PySspace and STARS: Space Time Analysis of Regional Systems®. Both will continue to exist and exploit a common library of functions.

PySspace is an open source software development effort that is part of PySAL to implement spatial statistical methods in general and spatial regression analysis in particular using Python and Numerical Python®. Current activities focus on a set of classes and methods to carry out diagnostics for spatial correlation in linear regression models and to estimate spatial lag and spatial error specifications.

The geospatial semantic web project builds on the GeoDa and PySAL software projects. A major distinction between this software and the web services is that these services are intended for automatic discovery by other services and not primarily targeted at human users. This requires the development of appropriate geospatial processing ontologies to facilitate the discovery and properly describe the data requirements, assumptions and computational functionality of the analytical services. This ontology is implemented in Web Ontology Language (OWL) and tested using web services programmed in Python and OWL-S (Semantic Markup for Web Services), SAWSDL (Semantic Annotations for Web Services Description Language), and the Web Service Modeling Ontology (WSMO) are compared as the framework to support discovery.

As soon as PySAL code is available, it will be announced on the OpenSpace mailing list and on this website. Spatial regression code in Python continues to be under development and will be announced in the same forums.
PySAL Code http://code.google.com/p/pysal

Project on Google Code (2009-08-14)

A library for exploratory spatial analysis and geocomputation

Code license: New BSD License
Labels: Python

Featured wiki pages: FileIO

Links: @geodacenter
Feeds: Project feeds
Groups: pysal-dev

Project owners: sjrey, lanselin
Project committers: andrewwinslow, sq/mtu
Project contributors: dfsb, dreamessence, mmocan, nhwang4, pedrovma, xinyue.ye
Outline

1. Origins
   - GeoDa and STARS
   - Goals and Organization

2. PySAL 1.0
   - Components
   - Illustration

3. Future
   - Organization
   - Invitation
Future

STARS

OpenGeoDa

PySpace

Web Services

Your Name Here
Python Spatial Analysis Library

pySAL

the open source for spatial analysis