

# Intro to Spatial Data Analysis in Python

Jenny Palomino (UC Berkeley)

FOSS4G NA 2015 Conference

March 10, 2015

# Quick Start: Spatial Data Analysis with Python

Spatial data types/formats

Python: what is it, how do you use it, and how can you get started?

Useful Open Source Python Spatial Packages

PySAL Example: Spatial Autocorrelation in Crime Data

Rasterio Example: Raster Calculation of Vegetation Index

Comparison of Python options to other geoprocessing options

Open source web options for publishing spatial data (including Python-based)

Online (Free!) Resources for Learning Python

# What is Spatial Data?

## Primary Data Types

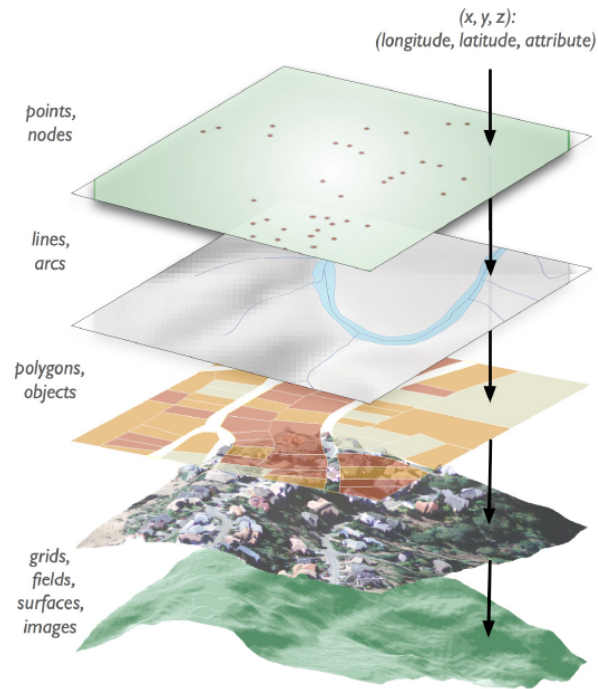
*vector*: point, line, polygon

*raster*: continuous (e.g. elevation) or discrete surfaces (e.g. land use)

## Common Data Formats

*vector*: shapefile, database geometry, tables (.dbf, .xlsx), KML, GeoJSON

*raster*: ASCII, GeoTIFF, JPEG2000, MrSID, database BLOB, HDF5



# What is Spatial Data?

## File Formats (data on hard drive)

shapefile, GeoTIFF, Table (.dbf, .xlsx), KML, HDF5

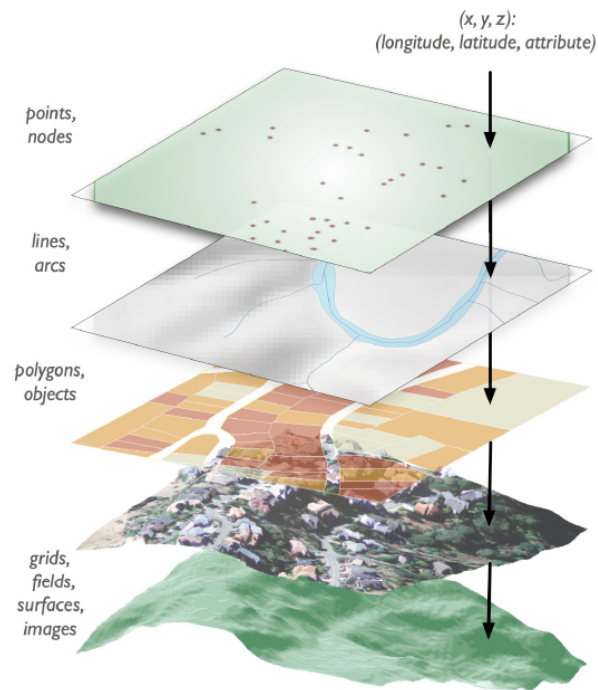
## Data Structures (data accessed via syntax)

GeoJSON, URL to dataset (HTTP to render WMS)

## Databases (data stored as geometry, BLOB, etc)

*Open Source:* PostgreSQL, MongoDB, SpatiaLite

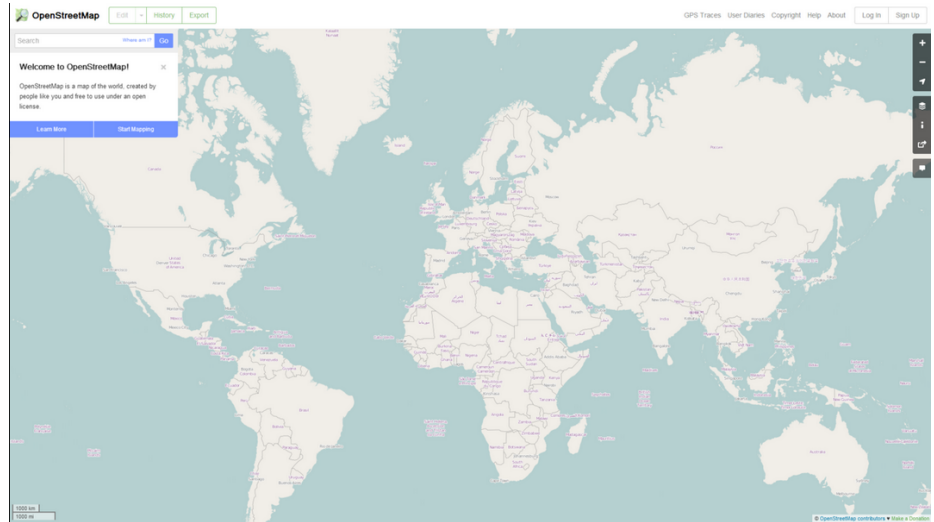
*Proprietary:* ESRI personal and file geodatabases, ESRI ArcSDE, Oracle Spatial, Microsoft SQL Server



# Map Projections: Translating 3D to 2D



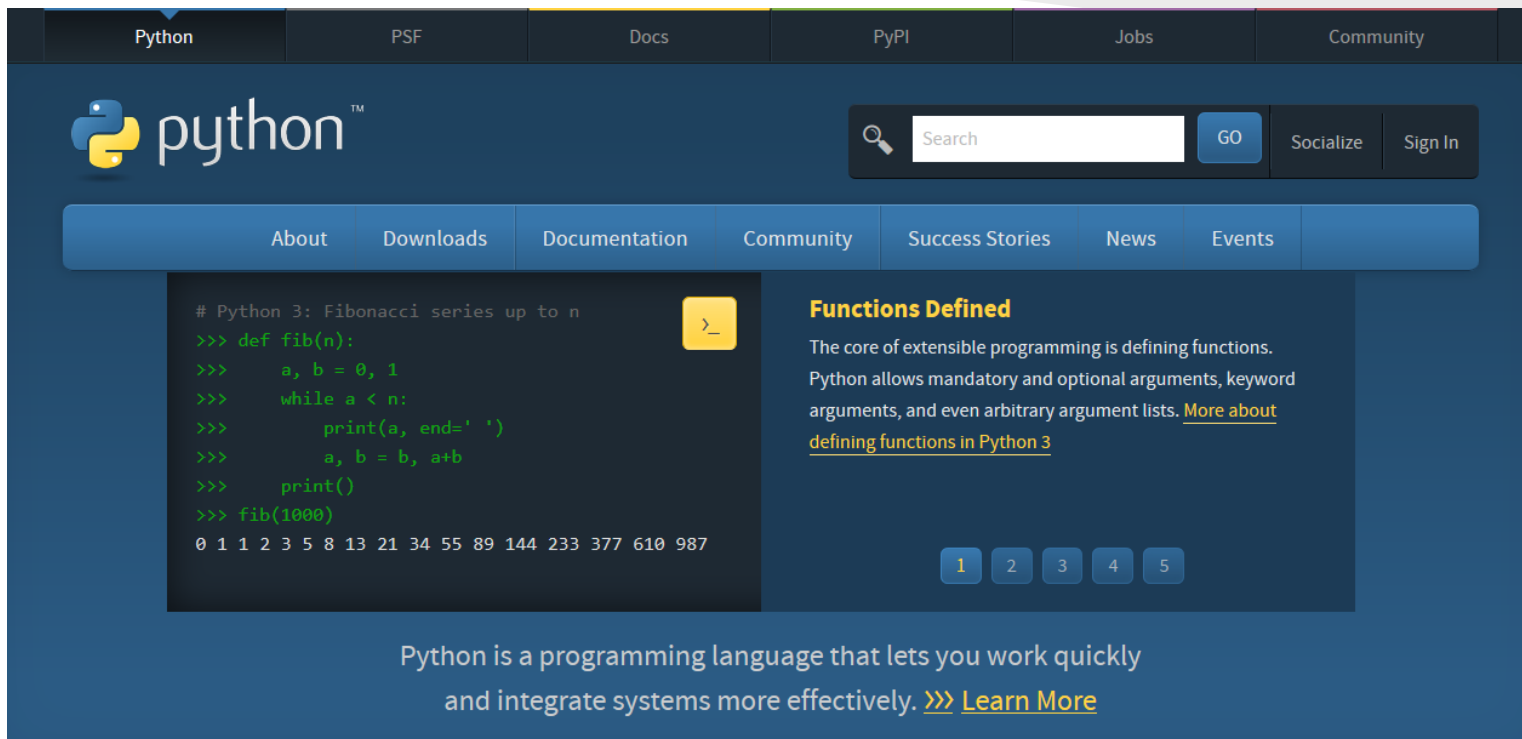
Visual Representation of UTM Zone 13



Web Mercator used in OpenStreetMap

<http://www.progonos.com/furuti/MapProj/Normal/TOC/cartTOC.html>  
[http://en.wikipedia.org/wiki/Web\\_Mercator](http://en.wikipedia.org/wiki/Web_Mercator)

# What is Python?



The image is a screenshot of the Python.org homepage. At the top, there is a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a large blue banner featuring the Python logo and the word 'python' in a stylized font. To the right of the logo is a search bar with a magnifying glass icon and a 'GO' button, followed by 'Socialize' and 'Sign In' links. Below the banner is a horizontal menu with links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area is divided into two columns. The left column contains a code snippet for a Fibonacci series generator, with a yellow terminal icon to its right. The right column has a section titled 'Functions Defined' with a brief description of Python's extensibility and a link to 'More about defining functions in Python 3'. At the bottom of the page, a blue footer contains the text 'Python is a programming language that lets you work quickly and integrate systems more effectively.' followed by a link to 'Learn More'.

Python

PSF

Docs

PyPI

Jobs

Community

python™

Search

GO

Socialize

Sign In

About

Downloads

Documentation

Community

Success Stories

News

Events

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

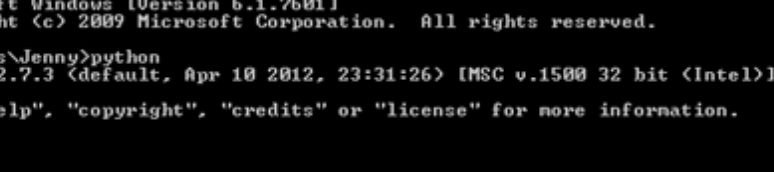
**Functions Defined**

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [>>> Learn More](#)

# Using Python: Traditional Install



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Jenny>python
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## Via Command Line (terminal or shell)

```
74 Toolbox Append Shapefiles Using Looping - Copy.py - C:\Users\Jenny\AppData\Local\Temp\Temp1...
File Edit Format Run Options Windows Help

#
# Append Shapefiles Using Looping.py
# A program that merges bird range shapefile data
# into one shapefile.
#

# Import required python libraries
import os, sys, string # needed to access OS file system and read file names
import arcpy # needed to access Arcpy library

# set workspace environment
arcpy.env.workspace = r"C:\Data\Birds" # Folder containing subfolders with in
arcpy.env.workspace = arcpy.GetParameterAsText(0)

#outshp = r"C:\Data\US\USBirds.shp" # The final output shapefile
outshp = arcpy.GetParameterAsText(1)
fcstring = ""

# set coordinate system
coordinateSystem = "GEOGCS['GCS_North_American_1983', DATUM['D_North_American_1983']]"

#clip_features = r"C:\Data\US\states.shp" # Extent that input shapefiles will be
clip_features = arcpy.GetParameterAsText(2)

xy_tolerance = ""
data_type = ""

# List all workspaces (subfolders that contain input shapefiles)
workspaces = arcpy.ListWorkspaces("", "ALL")

#Iterate through sub folders, set active workspace, find only polygons, define p
for workspace in workspaces:
    arcpy.env.workspace = workspace
    fcs = arcpy.ListFeatureClasses("", "Polygon")
    for fc in fcs:
        desc = arcpy.Describe(fc)
        if desc.ShapeType == "Polygon":
            arcpy.DefineProjection_management(fc, coordinateSystem)
            out_feature_class = r"C:\Data\Trash\\" + fc
            arcpy.Clip_analysis(fc, clip_features, out_feature_class, xy_tolerance, data_type)
```

Stand-alone script in a text editor (ex:IDLE)

# Using Python: Modern Options

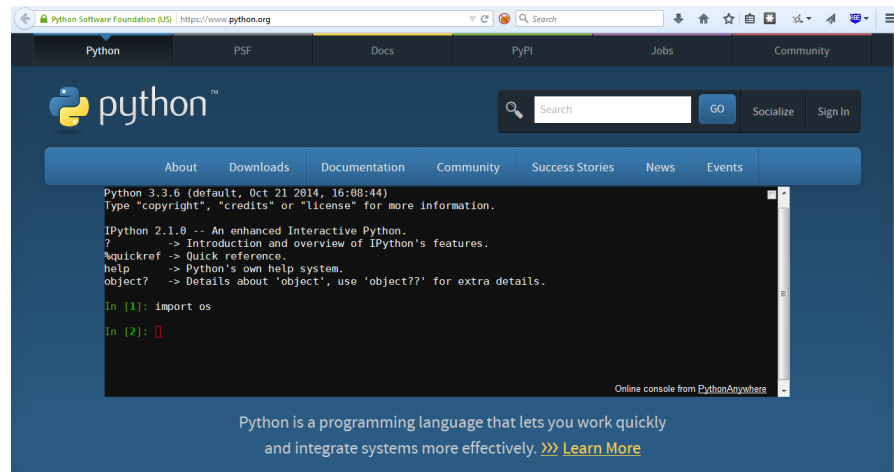
In [9]: `display(i)`

**IP[y]:** IPython  
Interactive Computing

In [3]: `from IPython.display import SVG`  
`SVG(filename='python-logo.svg')`

Out[3]:  python™

IPython  
(browser-based shell but still  
needs to be installed locally)



Python Anywhere  
(completely cloud-based)

<http://nbviewer.ipython.org/>  
<https://www.python.org/>



# Pre-packaged (and FREE) Python Distributions



## WinPython for Python 2.7:

- numpy 1.9
- scipy 0.15
- PySAL: not included
- pandas 0.15
- shapely: not included
- fiona: not included
- six 1.8
- Windows only

## Anaconda for Python 2.7:

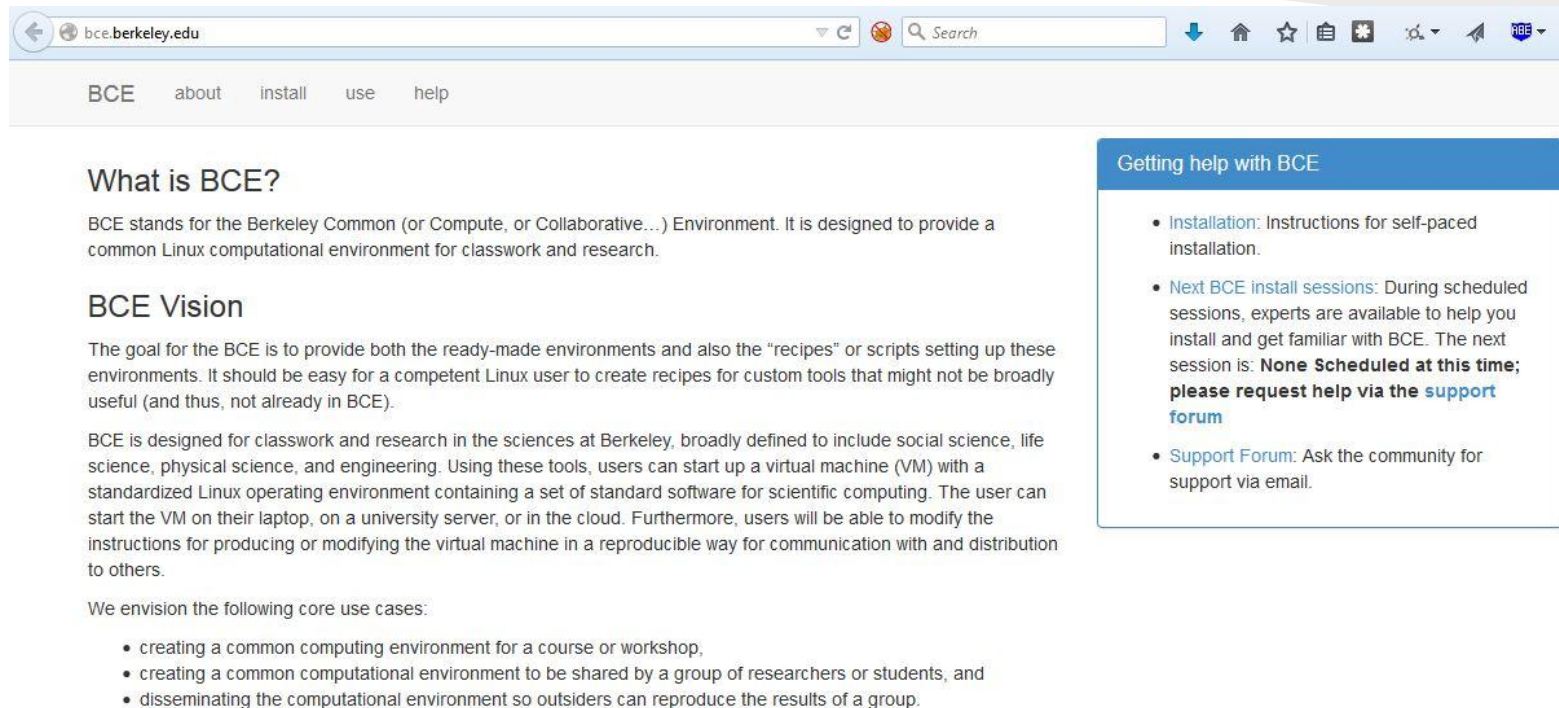
- numpy 1.9
- scipy 0.14
- PySAL 1.6
- pandas 0.14
- shapely: not included
- fiona: not included
- six 1.8
- Virtual Machine images
- Windows, Mac, Linux

## Enthought Canopy for Python 2.7:

- numpy 1.8
- scipy 0.15
- PySAL 1.7 (in academic option)
- pandas 0.15
- shapely: 1.5.1 (in academic option)
- fiona: 1.4.8 (in academic option)
- six 1.9
- Windows, Mac, Linux

Other Python distribution options listed at: <http://www.scipy.org/install.html>

# Berkeley Common Environment - Virtual Machine for Scientific Computing



The screenshot shows the Berkeley Common Environment (BCE) website. The browser address bar displays 'bce.berkeley.edu'. The navigation menu includes 'BCE', 'about', 'install', 'use', and 'help'. The main content area is titled 'What is BCE?' and describes the environment as a common Linux computational environment for classwork and research. It also includes a 'BCE Vision' section and a list of core use cases. A sidebar on the right, titled 'Getting help with BCE', provides links to installation instructions, next install sessions (noting that the next session is not scheduled at this time), and the support forum.

bce.berkeley.edu

BCE about install use help

## What is BCE?

BCE stands for the Berkeley Common (or Compute, or Collaborative...) Environment. It is designed to provide a common Linux computational environment for classwork and research.

## BCE Vision

The goal for the BCE is to provide both the ready-made environments and also the “recipes” or scripts setting up these environments. It should be easy for a competent Linux user to create recipes for custom tools that might not be broadly useful (and thus, not already in BCE).

BCE is designed for classwork and research in the sciences at Berkeley, broadly defined to include social science, life science, physical science, and engineering. Using these tools, users can start up a virtual machine (VM) with a standardized Linux operating environment containing a set of standard software for scientific computing. The user can start the VM on their laptop, on a university server, or in the cloud. Furthermore, users will be able to modify the instructions for producing or modifying the virtual machine in a reproducible way for communication with and distribution to others.

We envision the following core use cases:

- creating a common computing environment for a course or workshop,
- creating a common computational environment to be shared by a group of researchers or students, and
- disseminating the computational environment so outsiders can reproduce the results of a group.

### Getting help with BCE

- **Installation:** Instructions for self-paced installation.
- **Next BCE install sessions:** During scheduled sessions, experts are available to help you install and get familiar with BCE. The next session is: **None Scheduled at this time; please request help via the [support forum](#)**
- **Support Forum:** Ask the community for support via email.

# OSGeo Live - Virtual Machine for Open Source and Web GIS



# Useful Open Source Python Spatial Libraries

## *Data Handling:*

shapely

GDAL/OGR

pyQGIS

pyshp

pyproj

## *Analysis:*

shapely

numpy, scipy

pandas, GeoPandas

PySAL

Rasterio

scikit-learn, scikit-image

## *Plotting:*

matplotlib, prettyplotlib

descartes

cartopy

[https://github.com/SpatialPython/spatial\\_python/blob/master/packages.md](https://github.com/SpatialPython/spatial_python/blob/master/packages.md)  
<http://spatialdemography.org/essential-python-geospatial-libraries/>  
<http://carsonfarmer.com/2013/07/essential-python-geo-libraries/>

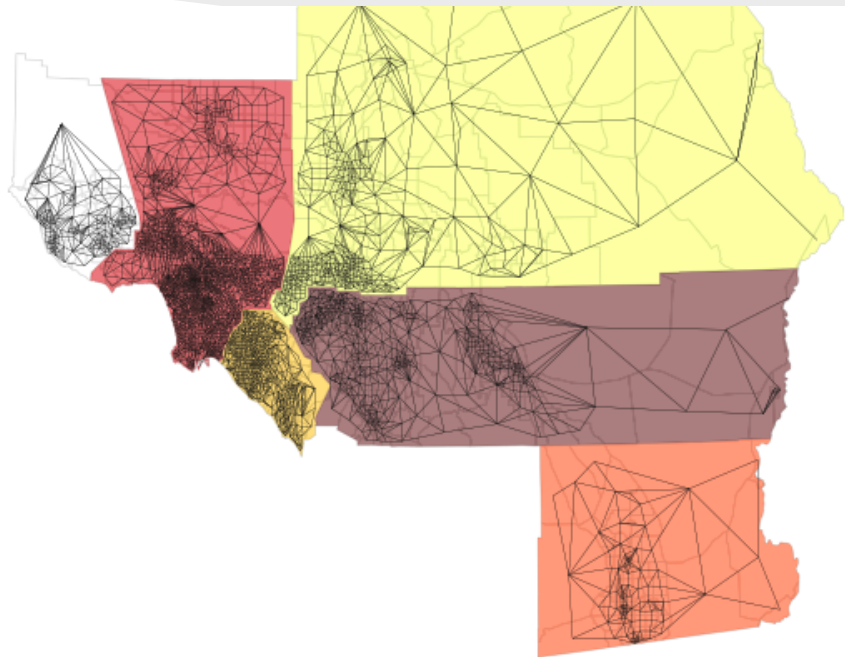
# PySAL - Python Spatial Analysis Library

## Vector and Raster Data Analysis

- Spatial Autocorrelation
- Spatial Econometrics
- Spatial Smoothing
- Regionalization
- Markov Chains

## Requires Python 2.6 or 2.7

- numpy (1.3 or later)
- scipy (0.11 or later)
- add shapely for more options

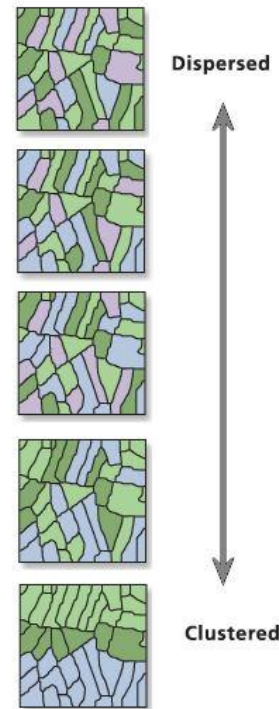


# What is Spatial Autocorrelation?

## Two Measures of Spatial Autocorrelation:

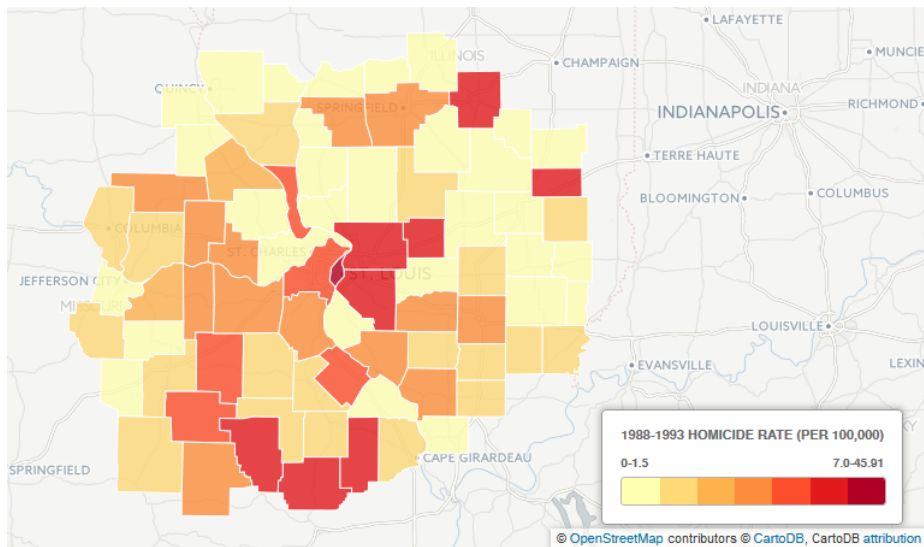
1. Global - quantifies clustering/dispersion across a region
  - a. values  $\sim 1.0$ : highly clustered
  - b. values  $\sim 0.0$ : no spatial autocorrelation
  - c. values  $\sim -1.0$ : highly dispersed
2. Local - identifies clusters (hot-spots) within the region

Example in presentation is based on Moran's I Global and Local Indicators (other indicators detailed here).

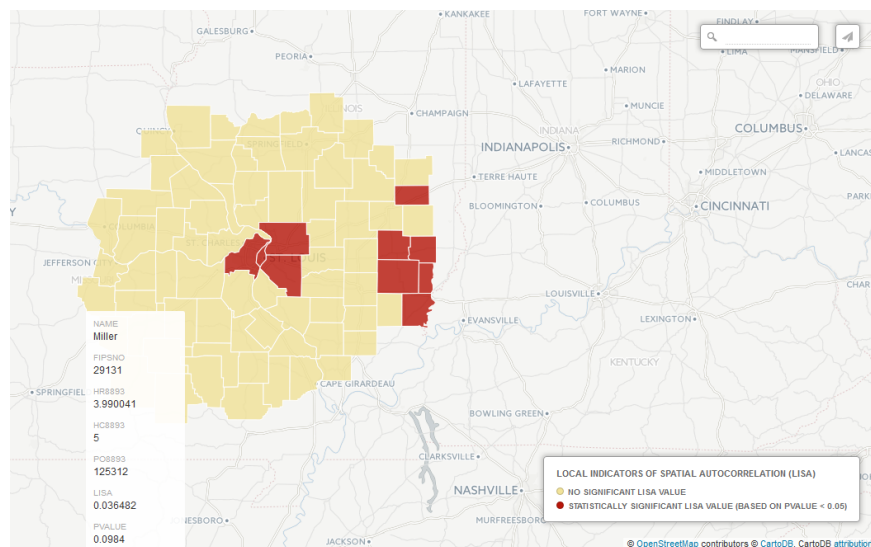


# Results of Spatial Autocorrelation in PySAL

Clustered with Global Moran's I = 0.24  
(CartoDB)



Three Clusters of Hot-spots  
(CartoDB)





# Why Use PySAL for this?

Compared to ArcMap, allows for more control over inputs and more robust testing of the data by including more advanced statistical methods:

- permutations of data (multiple runs of the random distribution)
- choice between one-tailed and two-tailed tests
- adjustments for populations of different sizes

Compared to R, leverage the benefits of working within Python:

- Object-oriented programming (advanced PySAL examples)
- Easily embed within desktop or web applications
- Easily work with Database Systems (e.g. PostgreSQL/PostGIS)
- Work with other useful Python packages (e.g. numpy, scipy, shapely)



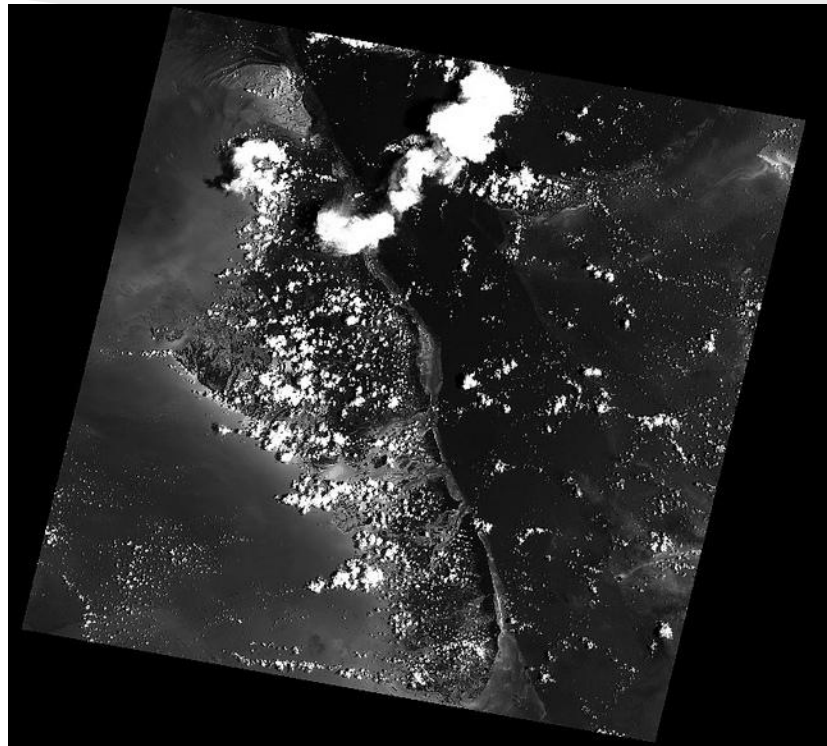
# Rasterio

## Raster Geoprocessing and Data Analysis

- Raster calculation across bands
- Raster stacking and merging
- Histograms and color maps
- Raster file conversions

## Requires Python 2.7, 3.3 or 3.4

- GDAL (1.9 or later)
- C compiler (more info)
- numpy (1.7 or later)
- enum34
- cligj
- affine (1.0 or later)



# Normalized Difference Vegetation Index (NDVI)

$$\text{NDVI} = \frac{(\text{Near Infrared Band} - \text{Red Band})}{(\text{Near Infrared Band} + \text{Red Band})}$$

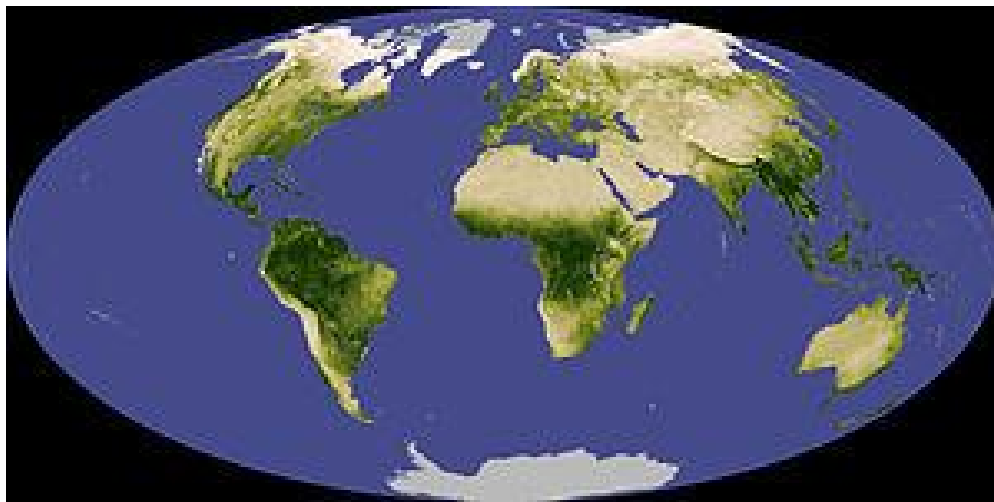
Values range from -1.0 to 1.0:

water ~ -1.0

barren area ~ 0.0

shrub/grass ~ 0.2-0.4

forest ~ 1.0

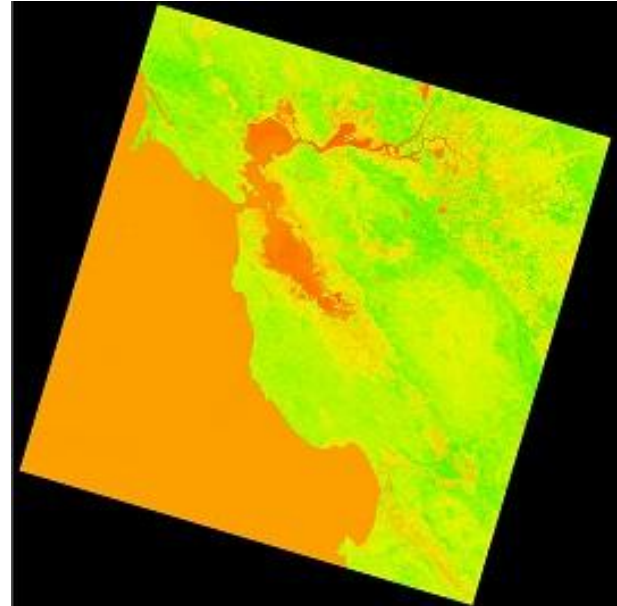


# Results of Raster Calculation in Rasterio

GeoTIFF of Landsat 8 image for the San Francisco Bay Area



NDVI result from  $\sim -1.0$  (red/orange) to  $\sim 1.0$  (green = vegetation)



# GeoPandas: Pandas + Shapely

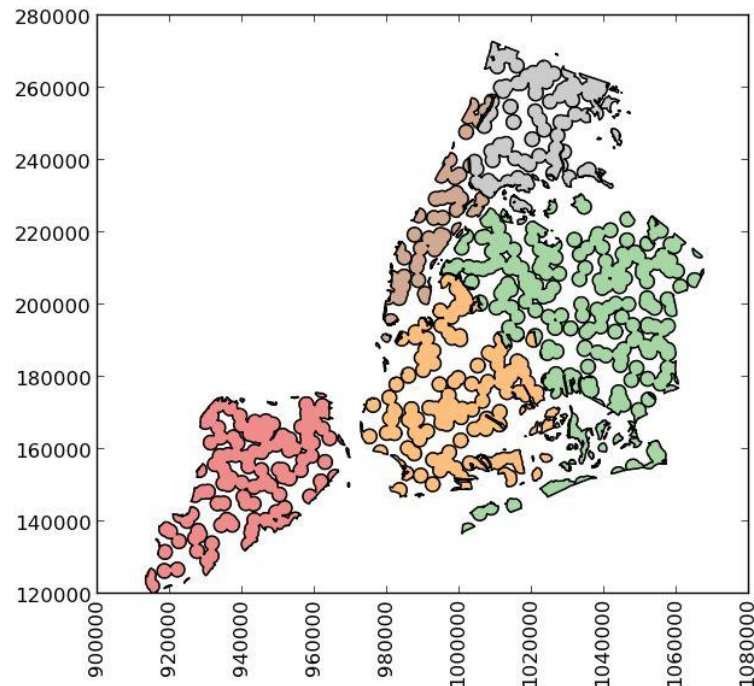
## Vector Geoprocessing

- buffer
- intersect
- union
- difference

## Requires Python 2.6, 2.7 or 3.2+

- numpy
- pandas (0.13 or later)
- shapely
- fiona
- six

```
>>> holes = boros['geometry'].intersection(mp)
```



# Comparing Python to other Spatial tools

	<i>ArcDesktop Suite</i>	<i>Open Source GIS (e. g., QGIS, uDig)</i>	<i>Python</i>	<i><u>R Spatial</u></i>
<i>Setting Up Working Environment</i>	Easy Install; Very clear GUI including Model Builder; restricted to Windows	<b>Easy Install; Relatively clear GUIs; OS independent</b>	Often many dependencies for installation of packages; (too?) many GUI and text editor options; OS independent	Requires installation of packages, but relatively easy; R Studio is nice GUI; OS independent
<i>Spatial Databases</i>	Needs ArcSDE for DBMS	work with major open source DBMS	<b>Works well with major open source DBMS</b>	Can work with major open source DBMS
<i>Analyzing Data</i>	Good for beginners and advanced users who use ArcPy; not much statistical power but solid data handling	Good for beginners; Grass allows more advanced analysis	<b>Statistical platform of choice for most sciences; handles “Big Data” the best</b>	<b>Statistical platform of choice for env and ecological sciences</b>
<i>Making a Map</i>	<b>Easiest option for Beginners</b>	Very Good option for Beginners	Requires willingness to learn programming fundamentals	Good for scientists already using R
<i>Publishing Data (web)</i>	ArcGIS Server: expensive but really easy to use	Doesn't have built-in option	<b>Web full of options</b> (details on next slide)	<u>R Shiny</u> is getting a lot of attention

# Open source options to publish spatial data

Easy Free Options: [CartoDB](#), [Google Maps API](#) and [Fusion Tables](#), [ArcGIS Online](#) and [ArcGIS Open Data](#) (both open to non-ESRI license holders)

Python-based: [GeoDjango](#), [Kartograph](#), [Mapnik](#) (C++ with Python bindings), [Mapserver](#) and [Open Street Map API](#) (support many languages including Python), [Flask](#) in combination with spatial database like MongoDB

JavaScript-based: [Kartograph](#), [Leaflet](#), [PolyMaps](#), [MapBox](#), [OpenLayers](#)

Spatial Data Servers and Suite of Tools: [Geoserver](#), [OpenGeoSuite](#)

Spatial Databases: [PostGIS](#), [MongoDB](#), [Spatialite](#)

# OS Geo Live - Virtual Machine for Open Source and Web GIS





# Online (and free!) Resources for Python

*Code Academy (programming tutorials):* <http://www.codecademy.com/>

*Coursera (full courses):* <https://www.coursera.org/courses?query=python>

*Python wiki pages:* <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>  
<https://docs.python.org/2/tutorial/>

*Python at Berkeley (DLab):* [http://python.berkeley.edu/learning\\_resources.html](http://python.berkeley.edu/learning_resources.html)

*Python Books and Training:* <http://pythonbooks.revolunet.com/>  
<http://www.learnpython.org/>

*ArcPy tutorials from ESRI:* <http://training.esri.com/gateway/index.cfm?fa=catalog.webCourseDetail&courseid=2520>

<http://training.esri.com/gateway/index.cfm?fa=catalog.webCourseDetail&courseid=2523>



# UC Berkeley: Spatial Data Science Bootcamp

*As technology is rapidly changing, the goal is not to teach a specific suite of tools but rather to teach participants how to develop and refine repeatable and testable workflows for spatial data using common standard programming practices.*

## Set Up Your Environment

- Virtual machine environments (Linux-based)
- Spatial databases (PostgreSQL/PostGIS) with multi-user editing and versioning (GeoGig)

## Wrangle Data

- APIs (Google Maps, OpenStreetMap)
- Modern data formats and tools (GeoJSON, GDAL)

## Analyze Data

- Vector-based analysis using ArcPy and PySAL (within IPython)
- Raster-based analysis using Rasterio (within IPython) and R

## Visualize and Publish Data

- Web-based visualizations (D3)
- Map publication (geostack, CartoDB)

Next Session: May 20-22, 2015 on UC Berkeley campus  
<http://iep.berkeley.edu/spatialdatascience>

# Questions or Comments?

[jpalomino@berkeley.edu](mailto:jpalomino@berkeley.edu)

# Evaluate the sessions

Sign in: [2015.foss4g-na.org/](http://2015.foss4g-na.org/)

