Python: Working With Scientific Data

Shaun Walbridge Kevin Butler

https://github.com/scw/scipy-devsummit-2015-talk

Handout PDF

High Quality PDF (28MB)

Python

Why Python?

- Accessible for new-comers, and the most taught first language in US universites
- Extensive package collection (56 thousand on PyPI), broad user-base
- Strong glue language used to bind together many environments, both open source and commercial
- Open source with liberal license — do what you want

. . .

- Brand new to Python? This talk may be challenging
- Resources include materials that for getting started

Release History

- ArcGIS 9.0 (2004, Python 2.1)
  - PythonWin COM bindings

```
import win32com
gp = win32com.client.dispatch("esriGeoProcessing.GpDispatch.1")
```

- ArcGIS 9.2 (2006, Python 2.4)
  - NumPy 1.0.3
  - Python-native module

```
import arcgisscripting
gp = arcgisscripting.create()
```

Release History

- ArcGIS 9.3 (2008, Python 2.5.1)
  - Python GP on Server
  - `gp = arcgisscripting.create(9.3)`
- ArcGIS 10.0 (2010, Python 2.6)
  - Fully integrated module: `import arcpy`
  - Python window
  - New extensions:
    - arcpy.sa
    - arcpy.mapping
    - arcpy.ga

Release History

- ArcGIS 10.1 (2012, Python 2.7)
  - Fast cursors: `arcpy.da.*`
  - Python Add-Ins and Python Toolboxes
  - Background Geoprocessing (64-bit)
  - matplotlib
- ArcGIS 10.3 (2014, Python 2.7.8)
  - Python 3.4 in Pro

  - NetCDF4
  - Python raster function, with a repository of examples using SciPy for on the fly visualizations

Release History

- Next:
  - SciPy stack
  - Package Management Enivonment (pip + the hard stuff)
  - Integration with R statistical language
- Move toward maintainable, reusable code and beyond the “one-off”

SciPy

Why SciPy?

- Most languages don’t support things useful for science, e.g.:
  - Vector primitives
  - Complex numbers
  - Statistics
- Object oriented programming isn’t always the right paradigm for analysis applications, but is the only way to go in many modern languages
- SciPy brings the pieces that matter for scientific problems to Python

Included SciPy

Package	KLOC	Contributors	Stars
matplotlib	63	312	2313
Nose	7	64	744
NumPy	84	299	1804
Pandas	112	349	4115

Package	KLOC	Contributors	Stars
SciPy	91	265	1528
SymPy	223	340	1981
Totals	580	1369	

1. An array object of arbitrary homogeneous items
2. Fast mathematical operations over arrays
3. Linear Algebra, Fourier Transforms, Random Number Generation

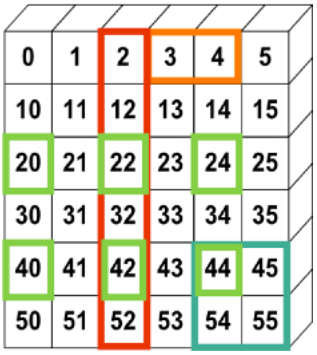


Figure 1:

SciPy Lectures, CC-BY

ArcGIS + NumPy

- ArcGIS and NumPy can interoperate on raster, table, and feature data.
- See Working with NumPy in ArcGIS
- In-memory data model. Example script to process by blocks if working with larger data.

ArcGIS + NumPy

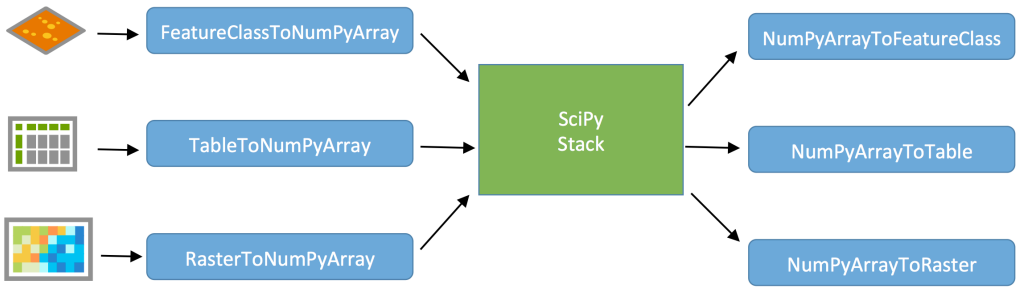


Figure 2:

- Plotting library and API for NumPy data

Computational methods for:

- Integration (scipy.integrate)
- Optimization (scipy.optimize)
- Interpolation (scipy.interpolate)
- Fourier Transforms (scipy.fftpack)
- Signal Processing (scipy.signal)
- Linear Algebra (scipy.linalg)

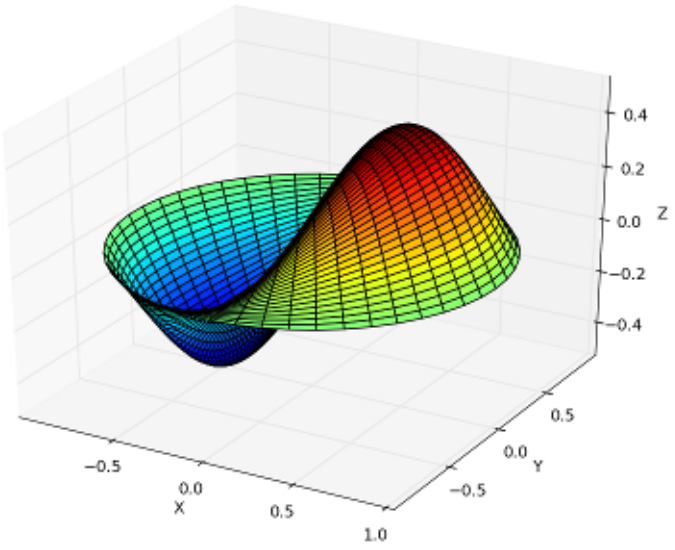


Figure 3:

- Spatial (scipy.spatial)
- **Statistics** (scipy.stats)
- **Multidimensional image processing** (scipy.ndimage)

Spatial is the tools across all of the domains of science, very general.

That said, can be useful in a variety of circumstances, e.g. `KDTree` for finding data quickly.

SciPy: Geometric Mean

- Calculating a geometric mean of an *entire raster* using SciPy (source)

$$\left(\prod_{i=1}^{n} a_i\right)^{1/n} = \sqrt[n]{a_1 \cdot a_2 \cdots a_n}$$

Figure 4:

```
import scipy.stats
rast_in = 'data/input_raster.tif'
rast_as_numpy_array = arcpy.RasterToNumPyArray(rast_in)
raster_geometric_mean = scipy.stats.stats.gmean(
    rast_as_numpy_array, axis=None)
```

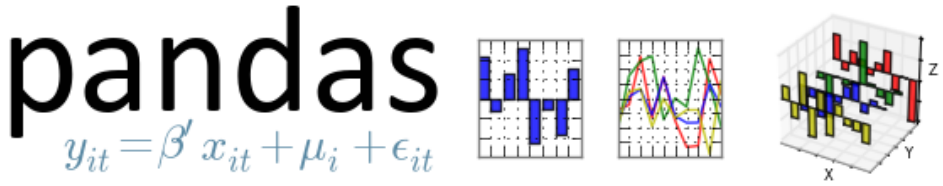
(Inspiration)

- **Pan**el **Da**ta — like R "data frames"
- Bring a robust data *analysis* workflow to Python

(Source)

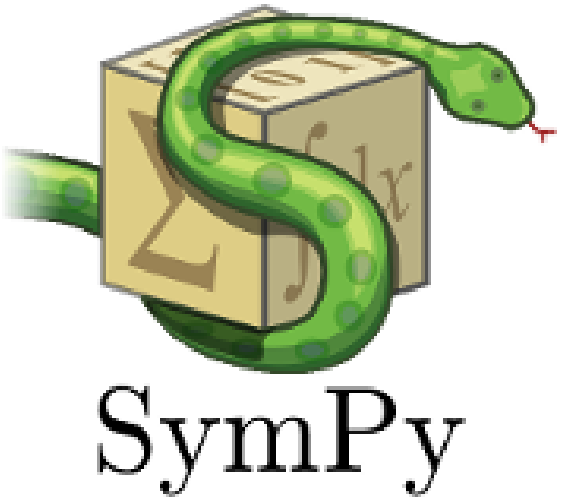
```
import pandas

data = pandas.read_csv('data/season-ratings.csv')
data.columns
```

```
Index([u'season', u'households', u'rank', u'tv_households', \
       u'net_indep', u'primetime_pct'], dtype='object')
```




```
majority_simpsons = data[data.primetime_pct > 50]
```

	season	households	tv_households	net_indep	primetime_pct
0	1	13.4m[41]	92.1	51.6	80.751174
1	2	12.2m[n2]	92.1	50.4	78.504673
2	3	12.0m[n3]	92.1	48.4	76.582278
3	4	12.1m[48]	93.1	46.2	72.755906
4	5	10.5m[n4]	93.1	46.5	72.093023
5	6	9.0m[50]	95.4	46.1	71.032357
6	7	8.0m[51]	95.9	46.6	70.713202
7	8	8.6m[52]	97.0	44.2	67.584098
8	9	9.1m[53]	98.0	42.3	64.383562
9	10	7.9m[54]	99.4	39.9	60.916031
10	11	8.2m[55]	100.8	38.1	57.466063
11	12	14.7m[56]	102.2	36.8	53.958944
12	13	12.4m[57]	105.5	35.0	51.094891



- A Computer Algebra System (CAS), solve math equations (source)

```
from sympy import *
x = symbol('x')
eq = Eq(x**3 + 2*x**2 + 4*x + 8, 0)


solve(eq, x)
```

$$x^3 + 2x^2 + 4x + 8 = 0$$

Figure 5:

$$\begin{bmatrix} -2, & -2i, & 2i \end{bmatrix}$$

Figure 6:

Where Can I Run This?

- Now:
  - ArcGIS Pro (64-bit) "Standalone Python Install for Pro"
    - Ships most of Scipy Stack (missing IPython)
  - NumPy: ArcGIS 9.2+, matplotlib: ArcGIS 10.1+
- Upcoming:
  - ArcGIS Desktop (32-bit), Background Geoprocessing (64-bit), Server (64-bit), Engine (32-bit)
  - IPython Included

Multidimensional Data

NetCDF4

- Fast, HDF5 and NetCDF4 read+write support, OPeNDAP
- Heirarchical data structures
- Widely used in meterology, oceanography, climate communities
- Easier: Multidimensional Toolbox, but can be useful

(Source)

```
import netCDF4
nc = netCDF4.Dataset('test.nc', 'r', format='NETCDF4')
```

```
print nc.file_format
# outputs: NETCDF4
nc.close()
```

- CF compliant data
- Fast, C-based access

Multi-D Improvements

- Multidimensional formats: HDF, GRIB, NetCDF
- Access via OPeNDAP, vector renderer, Raster Function Chaining
- An example which combines mutli-D with time
- Multi-D supported as WMS, and in Mosaic datasets (10.2.1+)

Demo: Benthic Terrain Modeler

Benthic Terrain Modeler

- A Python Add-in and Python toolbox for geomorphology
- Open source, can borrow code for your own projects: https://github.com/EsriOceans/btm
- Active community of users, primarily marine scientists, but also useful for other applications

Lightweight SciPy Integration

- Using `scipy.ndimage` to perform basic multiscale analysis
- Using `scipy.stats` to compute circular statistics

Lightweight SciPy Integration

Example source

```
import arcpy
import scipy.ndimage as nd
from matplotlib import pyplot as plt

ras = "data/input_raster.tif"
r = arcpy.RasterToNumPyArray(ras, "", 200, 200, 0)

fig = plt.figure(figsize=(10, 10))
```

Lightweight SciPy Integration

```
for i in xrange(25):
    size = (i+1) * 3
    print "running {}".format(size)
    med = nd.median_filter(r, size)

    a = fig.add_subplot(5, 5,i+1)
    plt.imshow(med, interpolation='nearest')
    a.set_title('{}x{}'.format(size, size))
    plt.axis('off')
    plt.subplots_adjust(hspace = 0.1)
    prev = med

plt.savefig("btm-scale-compare.png", bbox_inches='tight')
```

.

SciPy Statistics

- Break down aspect into `sin()` and `cos()` variables
- Aspect is a circular variable — without this 0 and 360 are opposites instead of being the same value

SciPy Statistics

Summary statistics from SciPy include circular statistics (source).

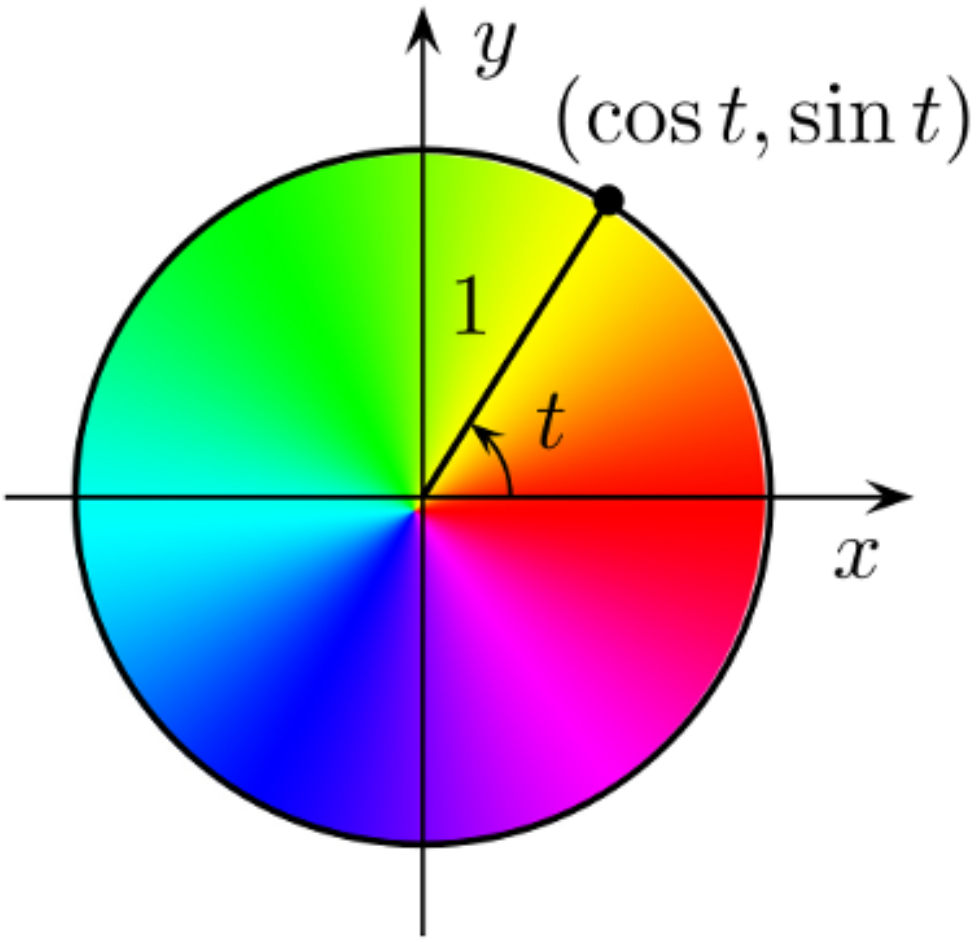


Figure 7:

```
import scipy.stats.morestats

ras = "data/aspect_raster.tif"
r = arcpy.RasterToNumPyArray(ras)

morestats.circmean(r)
morestats.circstd(r)
morestats.circvar(r)
```

Testing with Nose

- Nose — a Python framework for testing
- Tests improve your productivity, and create robust code
- Nose builds on `unittest` framework, extends it to make testing easy.
- Plugin architecture, includes a number of plugins and can be extended with third-party plugins.

Testing with Nose

An example test from `testMain.py` (full code, example):

```
class TestBpi(unittest.TestCase):
    def testBpiRun(self):
        input_raster = 'data/input_raster.tif'
        output_raster = 'test_run_bpi.tif'
        bpi.main(input_raster, 10, 30, output_raster)

        # Does our raster match the known mean?
        self.assertAlmostEqual(
            utils.raster_properties(output_raster, "MEAN"),
            0.295664335664)
```

Testing with Nose

Test specific BTM script:

```
cd tests/
nosetests -s testMain:TestBpi
```

Output:

```
----------------------------------------------------------------------
2 tests run in 7.1 seconds (2 tests passed)
----------------------------------------------------------------------
Ran 2 tests in 4.217s

OK
```

Testing with Nose

Supports many more options, and can integrate with tools like coverage to give you information about testing coverage, or the amount of code that you've added tests for, and what lines of code are missing from your tests:

```
nosetests --with-coverage
```

Testing with Nose

Coverage results:

```
Name                          Stmts   Miss  Cover   Missing
-----------------------------------------------------------
scripts.aspect                   24      4    83%   48-49, 53-54
scripts.bpi                      24      4    83%   57-58, 62-63
scripts.btm_model                39      4    90%   83-85, 89-90
scripts.classify                 90     20    78%   23, 30, 115-142
scripts.config                    4      0   100%
scripts.depth_statistics         39      4    90%   67-68, 72-73
scripts.ruggedness               49      4    92%   96-97, 101-102
scripts.slope                    18      4    78%   41-42, 46-47
scripts.standardize_bpi_grids    28      4    86%   57-58, 62-63
```

```
scripts.surface_to_planar     100     9    91%   179-180, 188-198
scripts.utils                  229     38    83%   37-51, 57-86
----------------------------------------------------------------------
TOTAL                          644     95    85

----------------------------------------------------------------------
45 tests run in 316.5 seconds (45 tests passed)
----------------------------------------------------------------------
Ran 45 tests in 316.091s

OK
```

from future import *

Opening Doors

- Machine learning (scikit-learn, scikit-image, …)
- Deep learning (theano, …)
- Bayesian statistics (PyMC, …)
  - Markov Chain Monte Carlo (MCMC)
- Frequentist statistics (statsmodels)

Resources

Other Sessions

- Python Raster Function: Custom On-the-fly Analysis
- Python: Working with Raster Data
- Python: Developing Geoprocessing Tools
- Integrating Open-source Statistical Packages with ArcGIS
- ArcGIS Pro: Map Automation with Python

New to Python

- Courses:
  - Programming for Everybody
  - Codecademy: Python Track
- Books:
  - Learn Python the Hard Way
  - How to Think Like a Computer Scientist

GIS Focused

- Python Scripting for ArcGIS
- ArcPy and ArcGIS - Geospatial Analysis with Python
- Python Developers GeoNet Community
- GIS Stackexchange

Scientific

Courses:

- Python Scientific Lecture Notes
- High Performance Scientific Computing
- Coding the Matrix: Linear Algebra through Computer Science Applications
- The Data Scientist's Toolbox

Scientific

Books:

- Free:
  - Probabilistic Programming & Bayesian Methods for Hackers
    * very compelling book on Bayesian methods in Python, uses SciPy + PyMC.
  - Kalman and Bayesian Filters in Python

Scientific

- Paid:
  - Coding the Matrix
    * How to use linear algebra and Python to solve amazing problems.
  - Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython
    * The cannonical book on Pandas and analysis.

Packages

Only require SciPy Stack:

- Scikit-learn:
  - Lecture material
  - Includes SVMs, can use those for image processing among other things…
- FilterPy, Kalman filtering and optimal estimation:
  - FilterPy on GitHub
- An extensive list of machine learning packages

Code

- ArcPy + SciPy on Github
- raster-functions
  - An open source collection of function chains to show how to do complex things using NumPy + scipy on the fly for visualization purposes
- statistics library with a handful of descriptive statistics included in Python 3.4.
- *TIP*: Want a codebase that runs in Python 2 and 3? Check out future, which helps maintain a single codebase that supports both. Includes the `futurize` script to initially a project written for one version.

Scientific ArcGIS Extensions

- Movement Ecology Tools for ArcGIS (ArcMET)
- Marine Geospatial Ecology Tools (MGET)
  - Combines Python, R, and MATLAB to solve a wide variety of problems
- SDMToolbox
  - species distribution & maximum entropy models
- Benthic Terrain Modeler
- Geospatial Modeling Environment
- CircuitScape

Conferences

- PyCon
  - The largest gathering of Pythonistas in the world
- SciPy
  - A meeting of Scientific Python users from all walks
- PyVideo
  - Talks from Python conferences around the world available freely online.
  - PyVideo GIS talks

Closing

Thanks

- Geoprocessing Team
- The many amazing contributors to the projects demonstrated here.
  - Get involved! All are on GitHub and happily accept contributions.

Rate This Session

www.esri.com/RateMyDevSummitSession