Intro to NumPy, SciPy, Matplotlib [*]

David Wright

February 27 2021

What We'll Be Covering Today

1. What are NumPy, SciPy, and Matplotlib?
2. Basic usage and functionality
3. Demos

What are NumPy, SciPy, and Matplotlib?

NumPy

- NumPy is the fundamental package for scientific computing in Python.
- Python library that provides the following:
  - Multidimensional array object (**ndarray**)
  - Various derived objects (such as masked arrays and matrices)
  - Assortment of routines for fast operations on arrays
    - routines include mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and more.

[*] This PDF document is an inferior version of an OER HTML page; free/libre Org mode source repository.

SciPy

- Built on NumPy
- Provides numerical routines, such as:
  - Numerical integration
  - Interpolation
  - Optimization
  - Linear algebra
  - Statistics

Matplotlib

[width=.9]figures/dave/matplotlib

- Library for creating visualizations in Python
  - Static,
  - Animated,
  - and Interactive visualizations

Basic Usage and Functionality

NumPy Basics

- Basic data type is **ndarray**

```
import numpy as np
x = np.array([[1,2,3],[4,5,6]])
print(type(x))
print(x.shape)
print(x)
```

```
<class 'numpy.ndarray'>
(2, 3)
[[1 2 3]
 [4 5 6]]
```

- Pre-compiled C code runs behind the scenes
  - Gives us speed and memory efficiency
- the arrays are **n-dimensional**
- `import as np` is the standard convention

NumPy Basics

As an example, I'll show how matrix multiplication can be done very easily with NumPy

```
import numpy as np
np.set_printoptions(suppress=True)
np.set_printoptions(precision=3)

x = np.array([1,0])
th = np.pi / 2
y = np.array([[np.cos(th), -np.sin(th)],
      [np.sin(th),  np.cos(th)]])
rot = np.matmul(y,x)
print(rot)
```

```
[0. 1.]
```

- **Note**: by default, the * operator performs element-wise multiplication on NumPy arrays

SciPy Basics

- SciPy is split into a number of **subpackages**

Subpackage	Description
cluster	Clustering algorithms
constants	Physical and mathematical constants
fftpack	Fast Fourier Transform routines
integrate	Integration and ordinary differential equation solvers
interpolate	Interpolation and smoothing splines
io	Input and Output
linalg	Linear Algebra
ndimage	N-dimensional image processing
odr	Orthogonal distance regression

SciPy Basics

Subpackage	Description
optimize	Optimization and root-finding routines
signal	Signal processing
sparse	Sparse matrices and associated routines
spatial	Spatial data structures and algorithms
special	Special functions
stats	Statistical distributions and functions

- Standard practice is to import only the subpackages you need

```
from scipy import optimize
```

Matplotlib Basics

- The basic usage is as follows

  ```
  import matplotlib.pyplot as plt
  plt.plot(#your-data)
  ```

- Matplotlib has many different plotting options
  - Histograms
  - Bar Charts
  - Errorbar
  - Scatter
  - 3D
  - Contours, and more

Python Demos

Visit the link below to get an online instance of a Jupyter Notebook with some demos.

- `https://mybinder.org/v2/gh/davecwright3/sps-computing-lectures/HEAD`

The End

Acknowledgements

- Snippets of Dr. Joseph Harrington's Python demos were used with his permission
- *ThinkPython* was used as a reference

Further Reading

- `https://greenteapress.com/wp/think-python-2e/`
- `https://diveintopython3.net/`