Los Angeles R Users' Group

NumPy and SciPy for Data Mining and Data Analysis Including iPython, SciKits, and matplotlib

Ryan R. Rosario

March 29, 2011

What is NumPy?

NumPy is the standard package for scientific and numerical computing in Python:

- powerful *N*-dimensional array object.
- sophisticated (broadcasting) functions.
- tools for integrating C/C++ and Fortran code (R is similar)
- useful linear algebra, Fourier transform and random number capabilities.

Additionally...

- data structures can be used as efficient multi-dimensional container of arbitrary data types.

So, What is SciPy?

Often (incorrectly) used as a synonym of NumPy.

- SciPy *depends on* NumPy's data structures.
- OSS for mathematics, science, and engineering.
- provides efficient numerical routines for numerical integration and optimization.

"NumPy provides the data structures, SciPy provides the application layer."

Getting NumPy

The ease of installation is highly dependent on OS and CPU.

- **Official** releases are recommended and available for
  1. 32 bit MacOS X
     - **Caveat:** Must use Python Python, not Apple Python.
  2. 32 bit Windows
- **Unofficial**/third-party binary releases for
  1. All Linux (use platform package system)
  2. 64 bit MacOS X (SciPy Superpack)
     - Requires Apple Python 2.6.1
  3. 64-bit Windows (Christoph Gohlke, UC Irvine)
- Source Code
- Enthought is to SciPy/Numpy as Revolution is to R.

Main download link is `http://www.scipy.org/Download`

Disclaimer

Time is even more limited than usual, so the goal is to illustrate what NumPy is, and what NumPy, SciPy and its colleagues can achieve. Some goals:

- Brief NumPy tutorial via demo.
- Uses of NumPy and SciPy, including SciKits.
- Brief data analysis demo with NumPy, SciPy and matplotlib.
- Concluding remarks.

NumPy Basics

The main data structure is a *homogeneous multidimensional array* – a table of elements **all of the same type**, indexed by a tuple of positive integers. Some examples are vectors, matrices, images and spreadsheets.

Why not just a use a list of lists? NumPy provides functions that operate on an entire array at once (like R), Python does not...really.

NumPy is full of syntax and time is limited, so let's jump straight into a demo...

NumPy Basics

Demo

Keep in Mind: Broadcasting vs. Recycling

In NumPy, if two arrays do not have the same length and we try to, say, add them, 1 is appended to the smaller array so that the dimensions match.

In R, if two vectors do not have the same length, the values in the shorter vector are recycled and a warning is generated. Personally, recycling seems more practical.

Keep in Mind: Indexing!

R starts its indices at 1 and ranges are inclusive (i.e. `seq(1,3) -> 1, 2, 3`)

NumPy (and Python) starts its indices at 0 and ranges are exclusive (i.e. `arange(1,3) -> 1, 2`).

Keep in Mind: Data Type Matters!

R does a good job of using the correct data type.

NumPy requires you to make decisions about data type. Using `int64` for 1 is typically wasteful.

ipython: a replacement CLI I

IP[y]:

IPython is the recommended CLI for NumPy/SciPy. I will use IPython in my demo.

1. an enhanced interactive Python shell.
2. an architecture for interactive parallel computing.

`ipython`: a replacement CLI II

Some advantages over the standard CLI:

1. tab completion for object attributes and filenames, auto parentheses and quotes for function calls.
2. simple object querying
   - `object_name?` prints details about an object.
   - `%pdoc` (docstring), `%pdef` (function definition), `%psource` (full source code).
3. `%run` to run a code file, much like `source` in R. Differs from `import` because the code file is reread each time `%run` is called. Also provides special flags for profiling and debugging.

`ipython`: a replacement CLI III

4. `%pdb` for spawning a `pdb` debug session.
5. **Output cache** saves the results of executing each line of code. `_`, `__`, `___` save the last three results.
6. **Input cache** allows re-execution of previously executed code using a snapshot of the original input data in variable `In`. (to rexecute lines 22 through 28, and line 34: `exec In[22:29] + In[34]`.
7. **Command history** with(out) line numbers using `%hist`. Can also toggle journaling with `%logstart`.
8. Execute system commands from within IPython as if you are in the terminal by prefixing with `!`. Use `!!`, `%sc`, `%sx` to get output into a Python variable (`var = !ls -la`). Use Python variables when calling the shell.

ipython: a replacement CLI IV

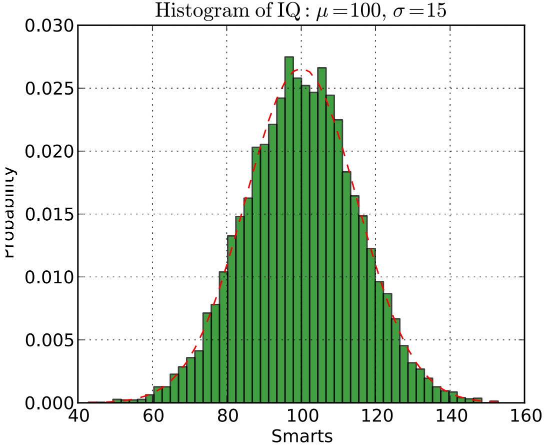
9. Easy access to the profiler, `%prun`.
10. Easy to integrate with `matplotlib`.
11. Custom profiles (options, modules to load, function definitions). Similar to R workspace.
12. Run other programs (that use pipes) and extract output (`gnuplot`).
13. Easily run doctests.
14. Lightweight version control.
15. Save/restore session, like in R (the authors specifically said "like R").

`matplotlib`: SciPy's Graphics System I

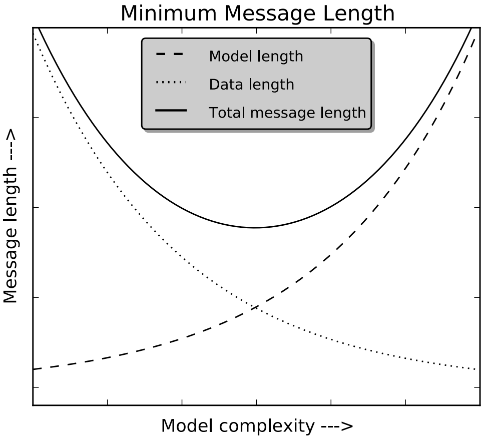
2D graphics library for Python that produces publication quality figures. Quality with respect to R is in the eye of the beholder.

- Graphics in R are much easier to construct, and require less code.
- Graphics in `matplotlib` may look nicer; require much more code.

`matplotlib`: SciPy's Graphics System II



matplotlib: SciPy's Graphics System III



`matplotlib`: SciPy's Graphics System IV

Show `matplotlib` gallery.

I will do a demo of `matplotlib` in the data analysis demo, if time.

Quick Linear Regression Demo

Demo here.

SciKits I

SciKits are modules or “plug-ins” for SciPy; they are too specialized to be included in SciPy itself. Below is a list of some relevant SciKits.

1. `statsmodels`: OLS, GLS, WLS, GLM, M-estimators for robust regression.
2. `timeseries`: time series analysis.
3. `bootstrap`: bootstrap error-estimation.
4. `learn`: machine learning and data mining

SciKits II

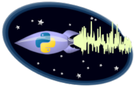
5. `sparse`: sparse matrices
6. `cuda`: Python interface to GPU powered libraries.
7. `image`: image processing
8. `optimization`: numerical optimization

For more information, see `http://scikits.appspot.com/scikits`.

Scientific Software Relying on SciPy I

Many exciting software packages rely on SciPy

Python **B**ased **R**einforcement Learning, **A**rtificial **I**ntelligence and **N**eural Network Library. Built around a neural networks kernel. Provides unsupervised learning and evolution.
`http://www.pybrain.org`



PyMC Pythonic Markov Chain Monte Carlo. MCMC, Gibbs sampling, diagnostics

PyML: SVM, nearest neighbor, ridge regression, multiclass methods, model selection, filtering, CV, ROC.

Scientific Software Relying on SciPy II

Monte: gradient based learning machines, logistic regression, conditional random fields.

MILK: SVM, k-NN, random forests, decision trees, feature selection, affinity propagation.

Modular toolkit for Data Processing: signal processing, PCA, ICA, manifold learning, factor analysis, regression, neural networks, mixture models.

Scientific Software Relying on SciPy III

PyEvolve: Genetic algorithms

Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently: GPUs, symbolic expressions, dynamic C code generation.

Orange: Data mining, visualization and machine learning. Full GUI and Python scripting.

Scientific Software Relying on SciPy IV

SAGE: a hodgepodge of open-source software designed to be one analysis platform.

Pros/Cons I

What NumPy/SciPy does better:

1. Much larger community (SciPy + Python).
2. Better, and cleaner graphics...slightly.
3. **Sits on atop a programming language rather than being one.**
4. Dictionary data type is very useful (Python).
5. Adheres to a system of thought ("Pythonic") that is more flexible.
6. Memory mapped files and extensive sparse matrix packages.
7. Easier to incorporate into a company/development workflow.
8. **Opinion:** Developers don't have to dive into C as soon as they do with R.

Pros/Cons II

What R does better:

1. One word `data.frame`.
2. Installation is simple and universal.
3. Application is self-contained (NumPy consists of several moving parts).
4. Very simple, user friendly syntax.
5. Graphics and analysis are fairly quick to code.
6. Much easier data input.
7. Handles missing values better.

The SciPy community is very useful, prolific and friendly. Project has very high morale. Some places to get help:

1. `http://www.numpy.org`: extensive documentation, examples and cookbook.
2. `http://www.scipy.org`: extensive documentation, examples and cookbook.
3. NumPy and SciPy mailing lists are helpful.
4. StackOverflow
5. IRC channel `#scipy`
6. Frequent sprints and hackathons.
7. SciPyCon (Austin, TX)
8. SoCalPiggies and LA Python (first meeting April 1!)

Keep in Touch!

My email: `ryan@stat.ucla.edu`

My blog: `http://www.bytemining.com`

Follow me on Twitter: `@datajunkie`

References

1. NumPy `http://www.numpy.org`
2. SciPy `http://www.scipy.org`
3. SciKits `http://scikits.appspot.com`
4. iPython `http://ipython.scipy.org/moin/`
5. matplotlib `http://matplotlib.sourceforge.net/`