Python™ vs. MATLAB® in Scientific Computing

Lijun Zhu

CeGP, Georgia Tech

April 22, 2016

Georgia Institute of Technology

Center for Energy & Geo Processing

CeGP

Outline

Disclaimers

- This sets of slides serves as an introduction to Python in scientific copmuting
- It is not intended to be a tutorial
- Many of the contents are subjective
- Useful info/webpages are given at the end of the presentation for future reference
- If interested, I can give a tutorial on any specific area

Outline

Motivation: a real-life scenario

- Seismic signal processing tasks
  - Download data from online data center
  - Process and plot the target data
  - Document code/algorithm for future reference
  - Save data and collaborate with others
- What I am looking for
  - Effortless data input/output
  - Easy and intuitive way to try some candidate algorithms
  - Generate great looking plots
  - Standard packaging and online user support
  - Good to transfer data and collaborate with other researchers

MATLAB® only solution

- Download dataset/station info from IRIS website
- Manually convert those info into MATLAB structure and use irisFetch.m to fetch data (Tedious)
- Process data and generate plots (plotseis, wiggle etc.)
- Improve plots for publishing (tightfig, tight_subplot etc.)
- Document code by MATLAB built-in docs and publish on personal website
- Save data in .MAT (later convert to other format such as .SAC, .SEGY, .SU using other third-party packages)

Mix MATLAB® with other tools

- Download dataset/station info from IRIS website
- Use AWK on UN*X system to generate a email template sent to IRIS serve
- Download data in .SEED format by FTP link from IRIS server
- Use SAC to convert .SEED to .SAC files
- Use MatSAC in MATLAB to load .SAC files
- Process data and generate plots in PNG/PDF format
- Post-process image to merge/modify plots into good appearance (Adobe Illustrator etc.)
- Document code by Latex generated PDF files/REAMME.me
- Save data in .MAT (later convert to other format such as .SAC, .SEGY, .SU using other third-party packages)

Python™ solution

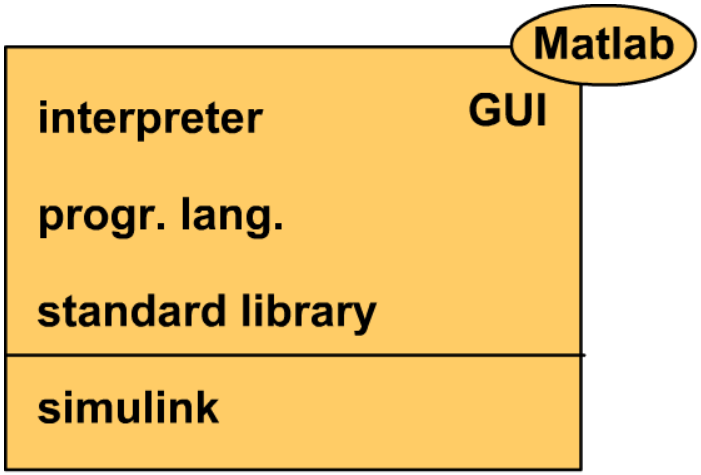
- Download dataset/station info from IRIS website and save in txt file
- Automatically(regular expression) load txt file into python and use ObsPy to request data from IRIS database (Easier)
- Process data by NumPy and SciPy, then generate plots by Matplotlib
- Improve plots by PyQtGraph etc.
- Document code by Sphinx and publish code in PyPI which is downloadable through pip
- Save data in the format that will be used to collaborate with others

Summary: a real-life scenario

- MATLAB® only solution
  - Unified development environment (+)
  - Optimized for matrix operation and linear algebra with little effort required (+)
  - Not ideal for some task (e.g. string manipulation) (−)
  - Limited to MATLAB tools (−)
- Mix MATLAB® with other tools
  - Use the right tool for the right task (+)
  - Full potential of optimization (+)
  - De-centralized development environment and difficult to package (−)
- Python™ solution
  - Unified development environment and packaging system for publishing code online
  - Full potential of optimization with incremental effort requirement

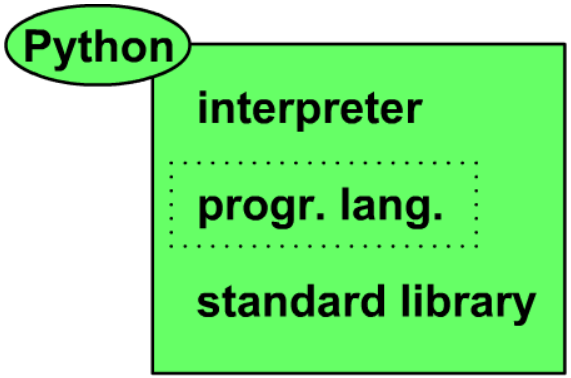
Outline

Python™ vs. MATLAB ®



Toolkits:
- image processing
- statistics
- optimization

etc.



GUIs:
- Wing
- Pype
- IEP
- Eclipse
- Komodo

etc.

Packages:
- numpy
- scipy
- pyOpenGl
- matplotlib
- visvis

etc.

Tools:
- pyrex
- py2exe
- py2app

etc.

Features comparison

Table 1: Basic and expendable feature comparison between MATLAB and Python

Features	MATLAB	Python
Interactive console	Built-in	IPython
IDE	Built-in	Sublime Text, Komodo, etc.
Matrix operation	Built-in	Numpy
Linear algebra	Built-in	Scipy
Plotting	Built-in	Matplotlib
Signal Processing	Toolbox	Scipy
Optimization	Toolbox	Scipy
Statistics	Toolbox	Pandas
Symbolic Math	Toolbox	Sympy
Expansion	C/C++, Fortran	C/C++, Fortran, R etc†.

- †: Python can be a wrapper for packages written in most other languages (Even MATLAB)

Interactive console

- MATLAB

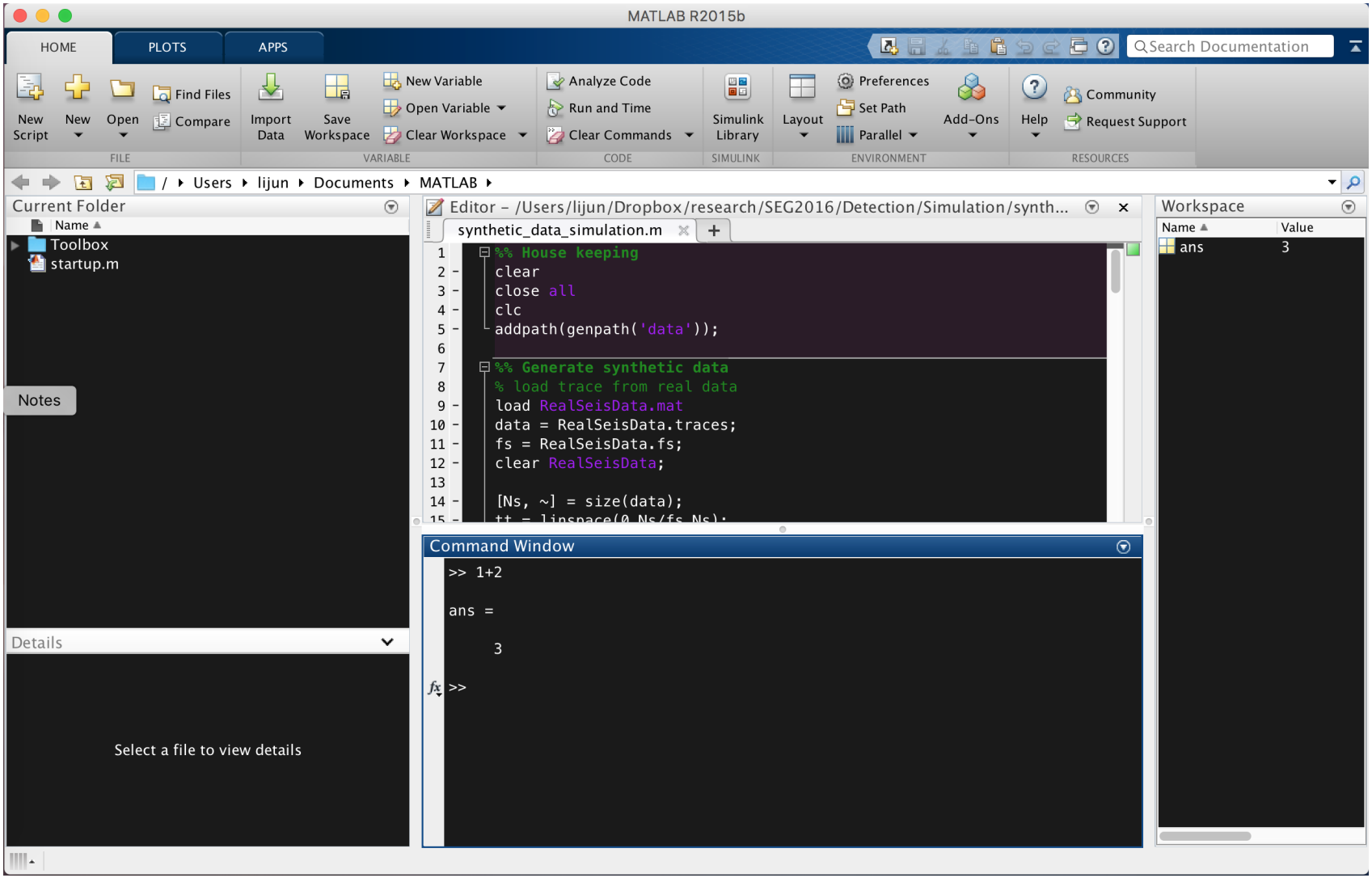
```
>> 1+2

ans =

     3

>>
```

- Python

```
Last login: Mon Apr 18 09:55:43 on ttys000
[lijun@Lijuns-MacBook-Pro:~$ ipython
Python 3.5.1 (default, Apr 10 2016, 07:35:31)
Type "copyright", "credits" or "license" for more information.

IPython 4.1.2 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

[In [1]: 1 + 2
Out[1]: 3

In [2]:
```

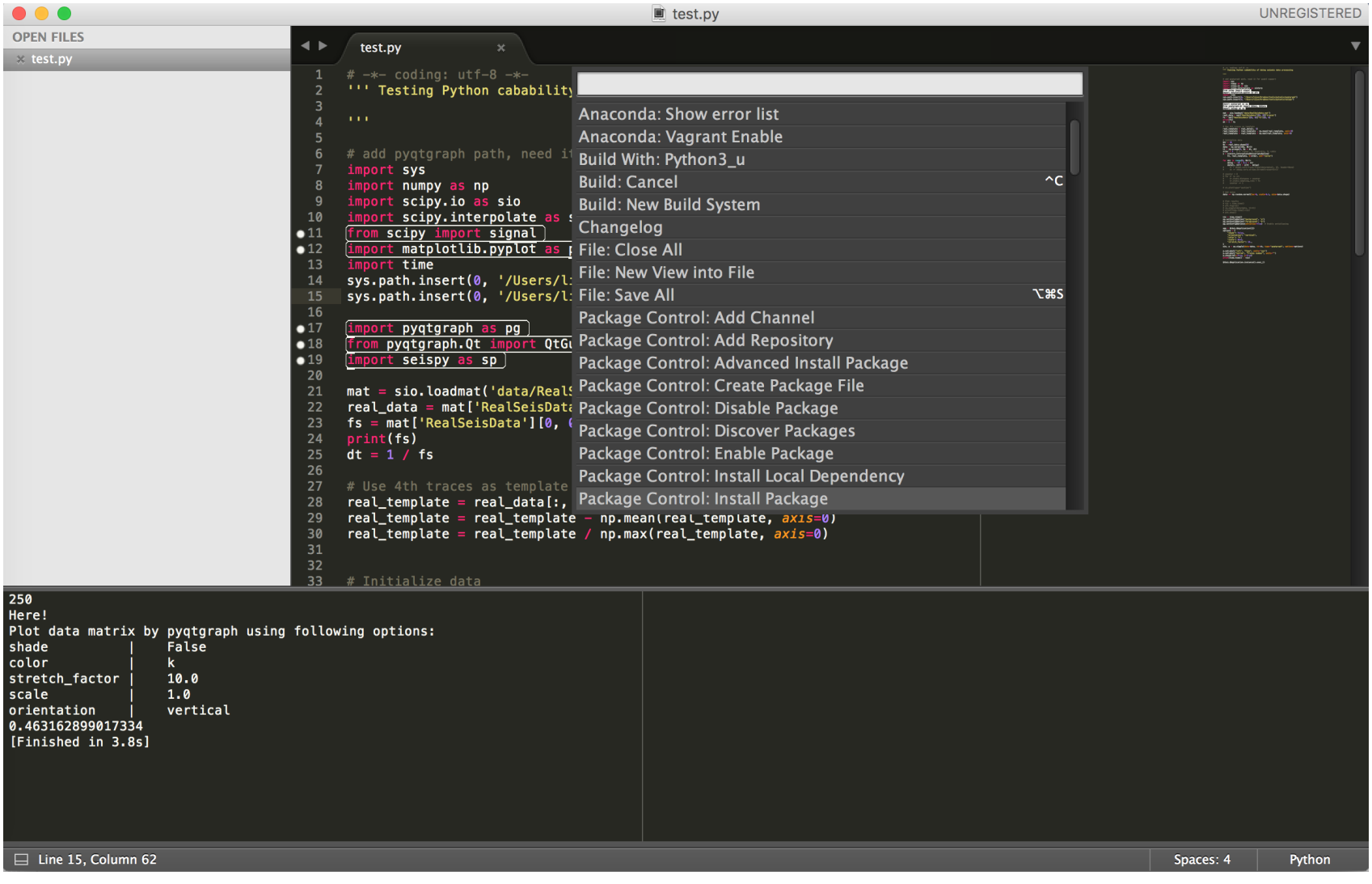
Integrated Development Environments(IDE): MATLAB

- Good UI and effective profiler
- Integrator console and file browser



Python: Sublime Text + Anaconda addon

- Nice and clean UI with most needed functonality
- Lack of a good profiler



Popular commercial Python IDE

- Komodo
  - Cross-Platform IDE for all your major languages, including Python, PHP, Go, Perl, Tcl, Ruby, NodeJS, HTML, CSS, JavaScript
  - The best IDE for Python with a high price tag ($295)
- Wing
  - Wing IDE Pro is a full-featured Python IDE designed for professional programmers. It includes powerful editor, code intelligence, refactoring, debugging, search, unit testing, project management, and revision control features.
  - Works and effective with lower price ($95)
- PyCharm
  - PyCharm provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django.
  - Good and usable IDE ($89/1st year, $71/2nd year, $53/3rd year)

Matrix operation

MATLAB

```
>> A = rand(3); B = eye(3);
>> A + B

ans =

    1.9649    0.9572    0.1419
    0.1576    1.4854    0.4218
    0.9706    0.8003    1.9157

>> A * B

ans =

    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
    0.9706    0.8003    0.9157

>> A'

ans =

    0.9649    0.1576    0.9706
    0.9572    0.4854    0.8003
    0.1419    0.4218    0.9157

>> A .* B

ans =

    0.9649         0         0
         0    0.4854         0
         0         0    0.9157
```

Python

```
In [1]: from numpy import *

In [2]: A = random.rand(3,3)

In [3]: B = eye(3)

In [4]: A + B
Out[4]:
array([[ 1.39990175,  0.35353724,  0.55751707],
       [ 0.87490555,  1.9834284 ,  0.7674989 ],
       [ 0.70632647,  0.72593449,  1.03217374]])

In [5]: dot(A,B)
Out[5]:
array([[ 0.39990175,  0.35353724,  0.55751707],
       [ 0.87490555,  0.9834284 ,  0.7674989 ],
       [ 0.70632647,  0.72593449,  0.03217374]])

In [6]: A.T
Out[6]:
array([[ 0.39990175,  0.87490555,  0.70632647],
       [ 0.35353724,  0.9834284 ,  0.72593449],
       [ 0.55751707,  0.7674989 ,  0.03217374]])

In [7]: A * B
Out[7]:
array([[ 0.39990175,  0.        ,  0.        ],
       [ 0.        ,  0.9834284 ,  0.        ],
       [ 0.        ,  0.        ,  0.03217374]])
```

Linear algebra

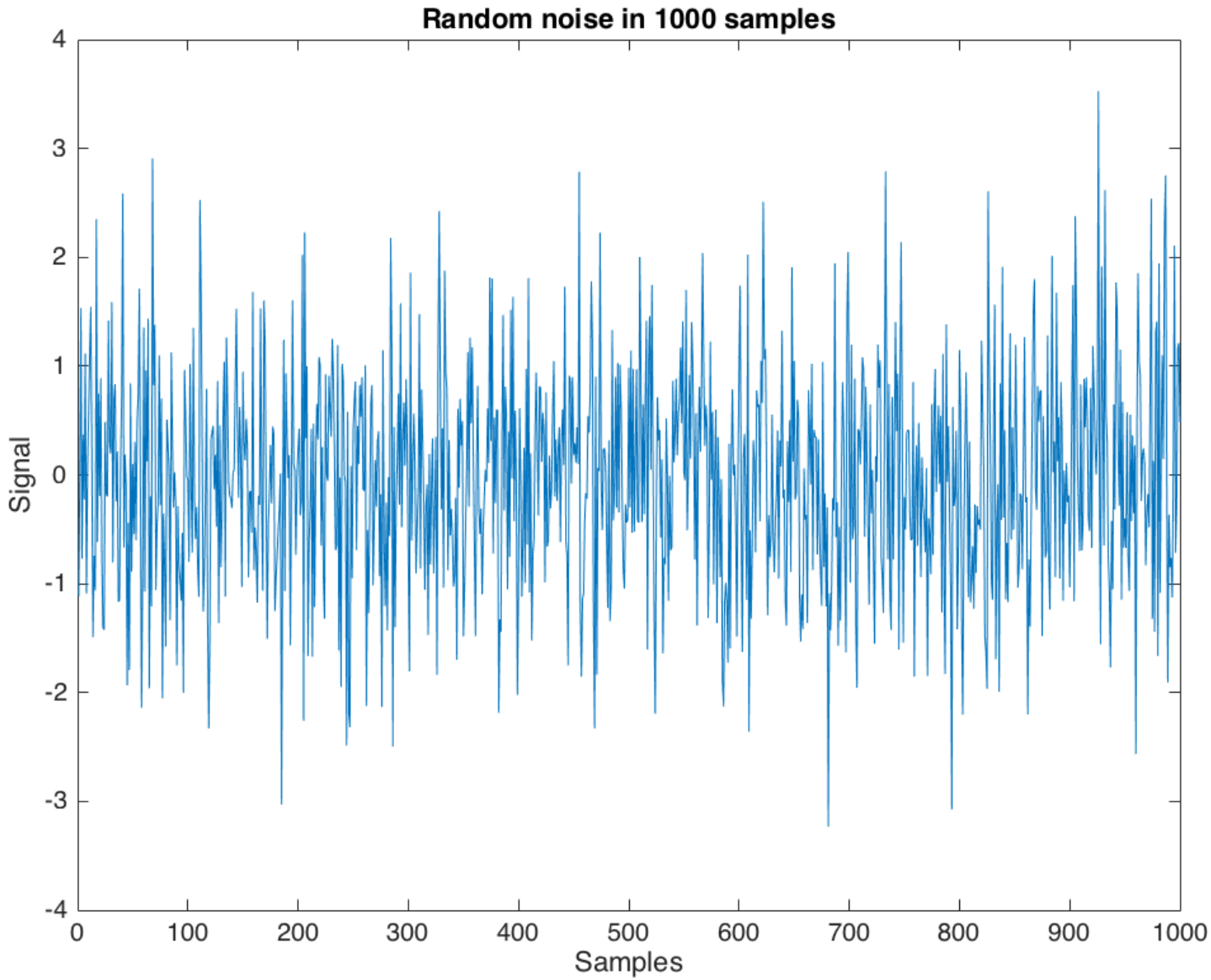
MATLAB

```
>> A = rand(3); b = rand(3,1);
>> x = A \ b

x =

    2.2575
    0.3263
   -1.7033

>> inv(A)

ans =

   30.0022  -26.5900   -4.4033
    6.4537   -6.4764    1.1218
  -35.4232   33.1663    3.9773

>> [V, D] = eig(A)

V =

   -0.6496   -0.6604   -0.3062
   -0.6543   -0.0906   -0.5276
   -0.3872    0.7455    0.7924


D =

    1.6817         0         0
         0    0.0319         0
         0         0   -0.2265
```

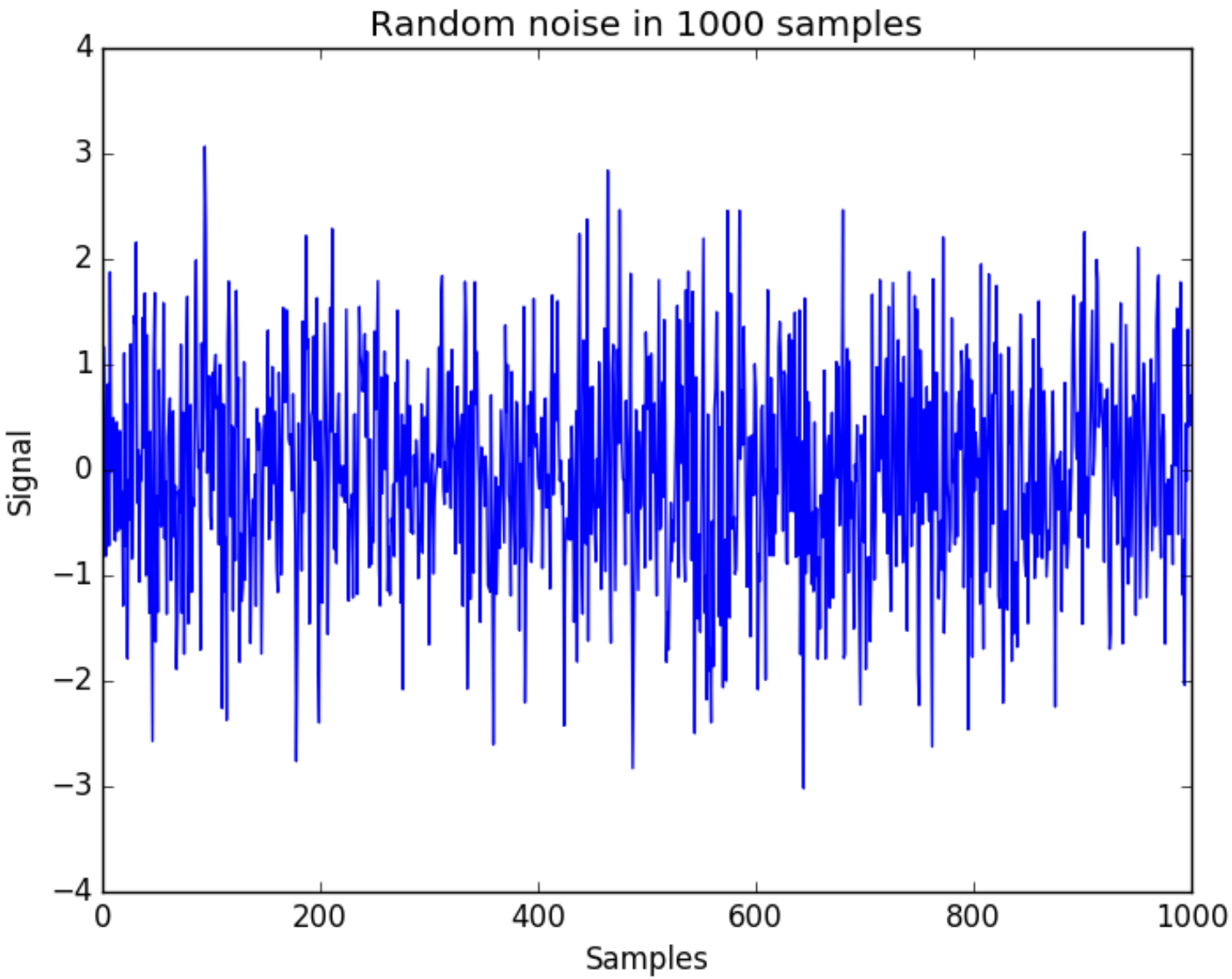
Python

```
[In [22]: A = random.rand(3,3)

[In [23]: b = random.rand(3)

[In [24]: linalg.solve(A,b)
 Out[24]: array([-1.92580779,  1.91305083,  1.42729985])

[In [25]: linalg.inv(A)
 Out[25]:
 array([[ 0.37713343,  2.53678956, -3.14390072],
        [ 2.14994234, -2.12139657,  2.31571086],
        [-3.23726686,  0.45446366,  2.17326778]])

[In [26]: linalg.eig(A)
 Out[26]:
 (array([ 1.45635949,  0.22390242, -0.21169082]),
  array([[-0.362874  , -0.56521975,  0.57210705],
         [-0.7423346 ,  0.10324115, -0.75588217],
         [-0.56325998,  0.81845458,  0.31833264]]))
```

Plotting(MATLAB)



Plotting(Python)



Signal Processing

MATLAB:

```
>> x = randn(1,10); h = randn(1,5);
>> y = conv(x,h)

y =

  Columns 1 through 7

   -0.2741   -0.9584    1.3466    1.8145    0.2053    4.9938    2.0834

  Columns 8 through 14

    0.1301    0.5937   -2.5208   -3.0337   -2.3839   -1.8772   -1.2787
```

Python:

```
In [4]: x = random.randn(10)

In [5]: h = random.randn(5)

In [6]: signal.convolve(x,h)
Out[6]:
array([ 0.36377455,  0.35152948,  0.08699449,  0.34243212,  1.94628262,
        2.28624853,  1.60599008,  2.46144222,  2.08663379,  0.35753257,
       -0.90937181, -0.08872919,  0.10801973, -0.01127863])
```

Optimization

- Find the minima of multi-dimensional Rosenbrock function
- MATLAB:

```
fun = @(x) sum(100.0*(x(2:end)-x(1:end-1).^2.0).^2.0 ...
    + (1-x(1:end-1)).^2.0);
x0 = [1.3, 0.7, 0.8, 1.9, 1.2];
options = optimoptions(@fminunc,'Display','iter',...
    'Algorithm','quasi-newton');
[x,fval,exitflag,output] = fminunc(fun,x0,options);
```

- Python:

```
import scipy.optimize as optimize

rosen = lambda x: sum(100.0 * (x[1:] - x[:-1]**2.0)**2.0
                      + (1 - x[:-1])**2.0)
x0 = np.array([1.3, 0.7, 0.8, 1.9, 1.2])
res = optimize.minimize(rosen, x0, method='BFGS',
                        options={'disp': True})
```

Summary: features

- Most MATLAB features has a corresponding package/module in Python
- Syntax of MATLAB and Numpy/Scipy are similar:
  - MATLAB syntax may be native to matrix operation
  - Python syntax is more similar to other programming language
- There is little thing in MATLAB that I cannot do in Python

Performance comparison between MATLAB and Numpy/Scipy

Items(1000×1000, sec)	MATLAB	Numpy/Scipy
Random matrix	1.43000e-02	3.53193e-02
Matrix summation	6.03940e-04	1.14631e-03
Matrix inverse	2.58000e-02	3.14017e-02
Matrix transpose	1.50000e-03	7.28061e-07
Matrix multiply	1.34000e-02	1.60881e-02
Solving linear system	1.41000e-02	1.63246e-02
Determinant	8.60000e-03	1.67287e-02
Norm	4.60000e-03	1.64028e-02
Eigenvalue Decompose	1.28600e+00	1.35907e+00
SVD	3.63800e-01	4.51565e-01
FFT	9.75997e-06	3.64929e-05/1.59512e-05
Spectrogram	4.62803e-03	2.55447e-03

- MATLAB comes Intel MKL hardware acceleration while Numpy/Scipy requires additional config/compilation (not included in above table)

Performance between MATLAB and Numpy/Scipy (cont.)

- In general, MATLAB(w/ MKL) is faster(no more than twice) than Numpy/Scipy(w/o MKL)
- Numpy/Scipy use fftpack instead of fftw due to license issues
- Numpy/Scipy matrix transpose is significantly faster than MATLAB due to data structure design; reason of slower norm is unknown
- Surprisingly, spectrogram is faster in Numpy/Scipy consider FFT has the opposite results

- In conclusion, Numpy/Scipy has comparable performance as MATLAB. Further optimization can be included if needed.

Case study: fast wiggle plot for large seismic dataset

- Demo

Case study: fast wiggle plot for large seismic dataset

- Demo
- MATLAB has better performance than Matplotlib

Case study: fast wiggle plot for large seismic dataset

- Demo
- MATLAB has better performance than Matplotlib
- Use PyQtGraph as an interface to Qt tools to optimize the graphic rendering

Case study: fast wiggle plot for large seismic dataset

- Demo
- MATLAB has better performance than Matplotlib
- Use PyQtGraph as an interface to Qt tools to optimize the graphic rendering
- The graphic rendering is faster, but background computation is slower than MATLAB (room to continue improve)

Case study: fast wiggle plot for large seismic dataset

- Demo
- MATLAB has better performance than Matplotlib
- Use PyQtGraph as an interface to Qt tools to optimize the graphic rendering
- The graphic rendering is faster, but background computation is slower than MATLAB (room to continue improve)
- Convert this wiggle() into a class to make passing options easier

Outline

Popularity: TIOBE Programming Community Index

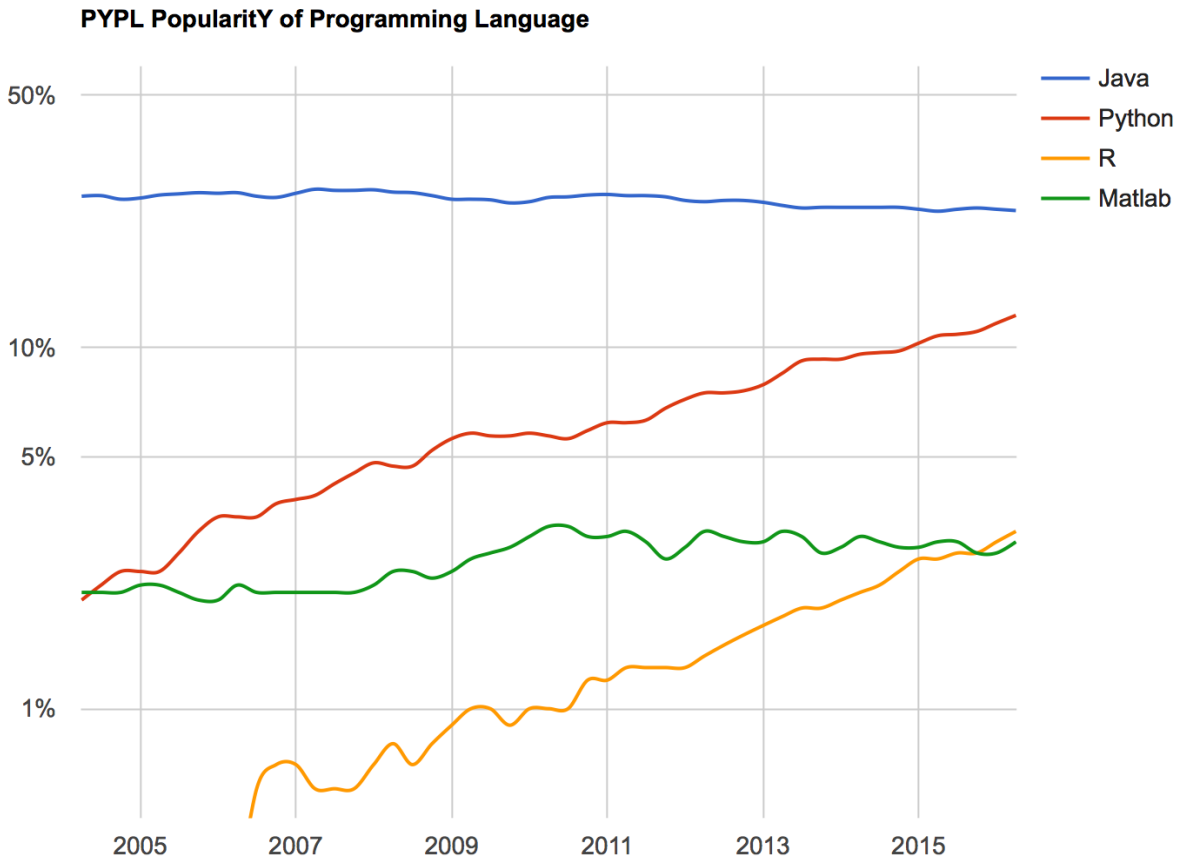
Apr 2016	Apr 2015	Change	Programming Language	Ratings	Change
1	1		Java	20.846%	+4.80%
2	2		C	13.905%	-1.84%
3	3		C++	5.918%	-1.04%
4	5	↑	C#	3.796%	-1.15%
5	8	↑	Python	3.330%	+0.64%
6	7	↑	PHP	2.994%	-0.02%
7	6	↓	JavaScript	2.566%	-0.73%
8	12	⇈	Perl	2.524%	+1.18%
9	18	⇈	Ruby	2.345%	+1.28%
10	10		Visual Basic .NET	2.273%	+0.15%
11	11		Delphi/Object Pascal	2.214%	+0.75%
12	29	⇈	Assembly language	2.193%	+1.54%
13	4	⇊	Objective-C	1.711%	-4.18%
14	9	⇊	Visual Basic	1.607%	-0.59%
15	24	⇈	Swift	1.478%	+0.60%
16	14	↓	MATLAB	1.344%	+0.08%
17	17		PL/SQL	1.314%	+0.20%
18	19	↑	R	1.266%	+0.24%

Popularity: PYPL

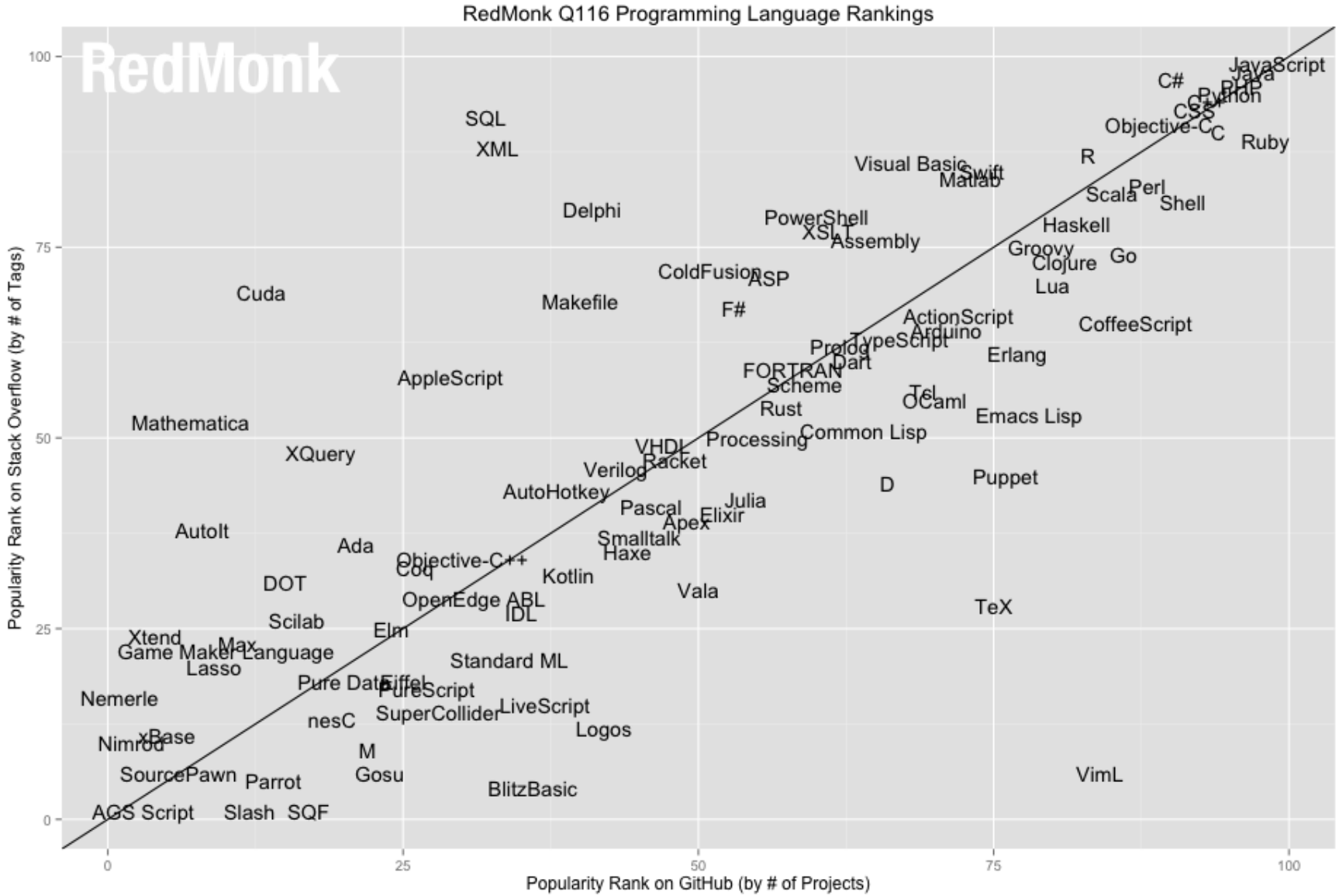
Worldwide, Apr 2016 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	24.0 %	-0.1 %
2	↑	Python	12.3 %	+1.7 %
3	↓	PHP	10.6 %	-1.0 %
4		C#	8.8 %	-0.2 %
5	↑↑	Javascript	7.5 %	+0.4 %
6	↓	C++	7.5 %	-0.3 %
7	↓	C	7.3 %	+0.2 %
8		Objective-C	4.8 %	-0.7 %
9	↑	R	3.1 %	+0.4 %
10	↑	Swift	3.0 %	+0.5 %
11	↓↓	Matlab	2.9 %	-0.1 %
12		Ruby	2.3 %	-0.3 %
13		Visual Basic	1.8 %	-0.4 %
14		VBA	1.5 %	+0.0 %
15		Perl	1.1 %	-0.2 %
16		Scala	0.9 %	+0.2 %
17	↑	lua	0.5 %	+0.1 %
18	↓	Delphi	0.4 %	-0.1 %

© Pierre Carbonnelle, 2015



Popularity: RedMonk Programming Language Rankings



Availability and compatibility

- Shortcomings of MATLAB®
  - Commercial software package (proprietary and can be expensive; but professionally maintained and optimized)
  - Implementation for functions/modules changes over different versions. Tough backward compatibility
  - Porting can be troublesome (MCR requires exact same version installed); MATLAB behaves a little different between Windows and Un*x
- How Python™ fixes them
  - Open source and free (personal and commercial use)
  - Community developed and maintained, possible bugs
  - Dedicated tools to even the gap between v.2 and v.3
  - ALMOST platform independent
  - Virtual environment is ideal for porting and collaboration

Programming language

- Language wise
  - MATLAB is historically a wrapper around LA/SP tools developed in C/Fortran. It's evolving into a programming language
  - Python is a high-level, general-purpose, interpreted, dynamic programming language. It can be used in scientific computing
- Object-oriented programming is natural in Python while can be awkward sometimes in MATLAB
- Functions in scripts are allowed Python but not MATLAB. Much few files to keep in development and better project structure.
- Data structure wise
  - MATLAB: array, cell, structure; matrix based computation
  - Python: list, set, dict, tuple; generator based computation

Readability

- MATLAB:

```
data = randn(100);
% Normalize data
data = bsxfun(@rdivide,data,std(data,[],1));
% alternatively
data = data ./ repmat(std(data,[],1),size(data,1),1);
```

- Python:

```
import numpy as np
data = np.random.normal(loc=0., scale=1., size=100)
# Normalize data
data = data / np.std(data, axis=0)
```

Readability (cont.)

- MATLAB is more intuitive for linear algebra operations
- MATLAB:

```
A = randn(3); b = randn(3,1)
x = A \ b
```

- Python:

```
A = np.random.randn(3, 3)
b = np.random.rand(3)
x = np.linalg.solve(A, b)
```

Readability (cont.)

- Python allow function definition in a script/module

```
def classify(values, boundary=0):
    '''Classifies values as being below
    (False) or above (True) a boundary.
    '''
    return [(True if v > boundary else False)
            for v in values]

# Call the above function
my_values = np.array([1])
classify(my_values, boundary=0.5)
```

- Those functions are usable from "Import" and make code generally more compact and orgainized
- Matlab requires lots of small files for function definition which propose "copy and paste" programming

Balance of High Level and Low Level Programming

- Use Cython to convert python module to C code (Demo)

Balance of High Level and Low Level Programming

- Use Cython to convert python module to C code (Demo)
- Use Python C/C++API to import complete C/C++code or library (like MEX in MATLAB)

Test frame work

```
import unittest


class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        # check that s.split fails when the separator is not a
        with self.assertRaises(TypeError):
            s.split(2)
```

Online resource and document

- File Exchange vs. PyPI
- List of functions (Hyperplot Tools) vs. Online documentation (PyQtGraph)
- Software install/update: Manually download vs pip
- Fragmental(MATLAB) vs centric(Python) package search experience
- Standard project and package system(Cookiecutter) with integration of Github, GitFlow, Sphinx, and PyPI

Downsides of Python

- Too many choices for packages, difficult to choose (ironic, but true)
- Many scientific computing packages are developed and distributed in MATLAB
- Good news: they can be ported to Python
  - CVX: CVXOPT
  - SPGL1: SPGL1_python_port

Outline

Useful packages for Python

- Python comes with a very small set of built-in module for basic operation and file handling
- Packages must-have for scientific computing
  - Numpy for matrix operation
  - Scipy for linear algebra, signal processing, optimization and etc.
  - Matplotlib for 2D plot
- Packages recommended for trying
  - pyFFTW for fast Fourier Transform
  - Seabron for statistical data visuallization
  - PyQtGraph for scientific graphics and GUI library to generate fast and interactive plots
  - SAGE aims to be a replacement for MATLAB
  - ObsPy is a Python framework for seismology (segy, sac i/o)
  - scikit-learn for machine learning

ObsPy

- ObsPy is an open-source project dedicated to provide a Python framework for processing seismological data.
- Handles a good variety of file types and databases with a large library
- Generate standard plots for natural earthquake publication

Seaborn

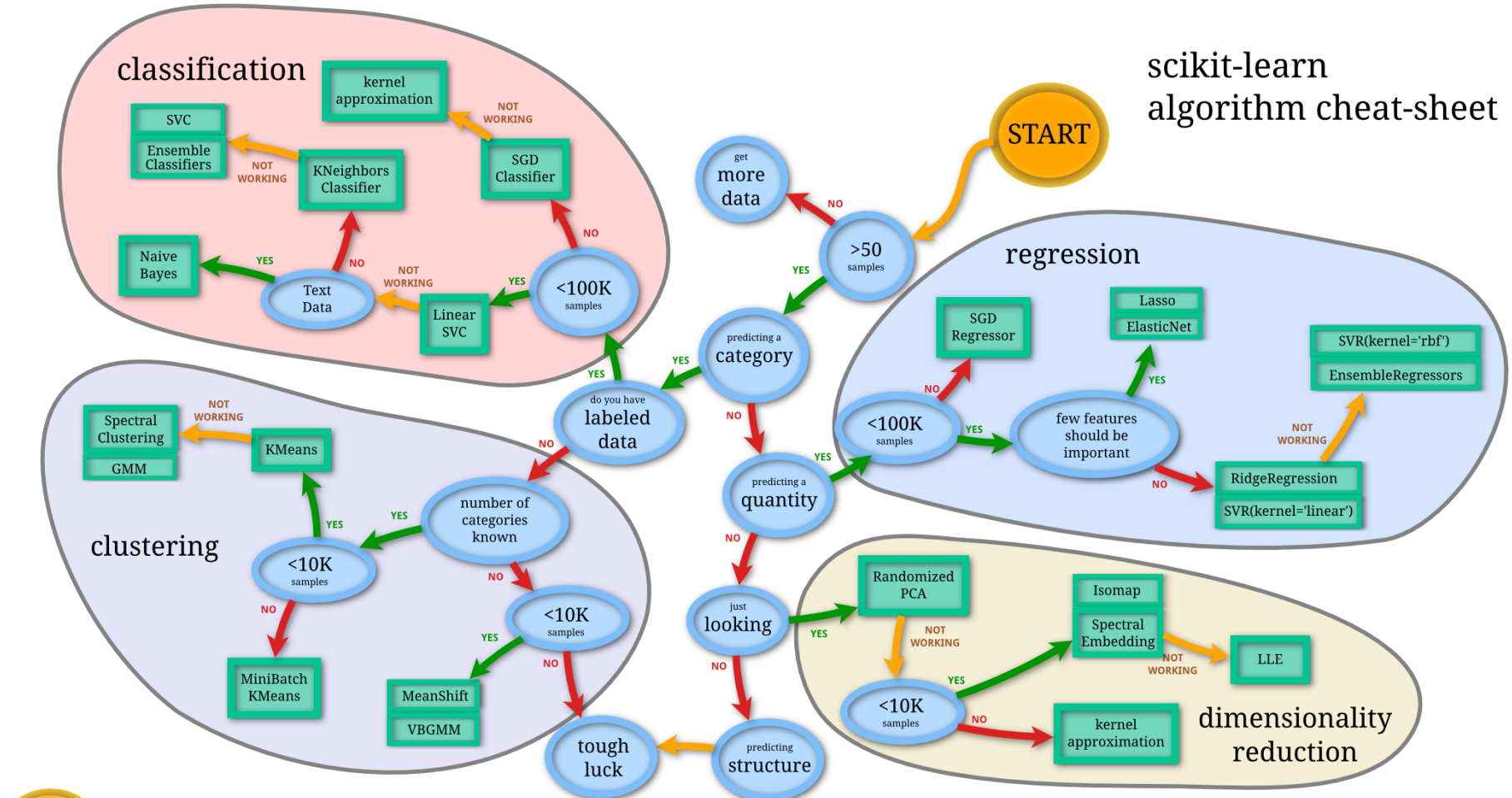
- Excellent lib for 2D plot of statistical data
- Most types of plot you can think of
- Easy install and setup

SageMath

- Their mission: create a viable free open source alternative to Magma, Maple, Mathematica and Matlab
- Popular in some mathematical area
- Includes: NumPy, SciPy, matplotlib, Sympy, Maxima, GAP, FLINT, R etc.

scikit-learn

- A great easy-to-start machine learning framework in Python
- Includes common methods for supervised learning, unsupervised learning, model selection etc.
- Good material for learning those ML algorithms.


scikit-learn
algorithm cheat-sheet
START
get more data
>50 samples
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
kernel approximation
SVC
Ensemble Classifiers
KNeighbors Classifier
SGD Classifier
Naive Bayes
Text Data
Linear SVC
<100K samples
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
<100K samples
few features should be important
RidgeRegression
SVR(kernel='linear')
clustering
Spectral Clustering
GMM
KMeans
number of categories known
<10K samples
MiniBatch KMeans
<10K samples
MeanShift
VBGMM
dimensionality reduction
Randomized PCA
Isomap
Spectral Embedding
LLE
<10K samples
kernel approximation
YES
NO
NOT WORKING
Back

OpenCV-Python

- OpenCV-Python is a library of Python bindings designed to solve computer vision problems
- OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation
- OpenCV-Python makes use of Numpy and can be easily integrated with SciPy and Matplotlib.

Complete programming solution + full-functional playground

- MATLAB has a mature IDE with sweet number of packages
- MATLAB is optimized for matrix operation and is ideal to test the water for new algorithms (Fancy scientific calculator with programming functionality)

- Python is more versatile with reasonable performance
- Python packages are easy to develop, deploy and maintain

- I see a future that using Python as the main development tools while keep MATLAB as a playground for quick algorithm verification

Reference: Python/MATLAB comparison

- Cyrille Rossant: Why use Python for scientific computing?
- Almar Klein: Python vs Matlab
- Steve Tjoa: I used Matlab. Now I use Python.
- Philippe Feldman: Eight Advantages of Python Over Matlab
- Vincent Noël: Bye Matlab, hello Python, thanks Sage
- Hoyt Koepke: 10 Reasons Python Rocks for Research (And a Few Reasons it Doesn't)

IRIS database and SAC tools

- IRIS DMC: (SAC) a general purpose interactive program designed for the study of sequential signals, especially time series dat
- IRIS DMC: (irisFetch.m) the Matlab library IRISFETCH allows seamless access to data stored within the IRIS-DMC
- Zhigang Peng: (MatSAC) Read/Write/Plot seismic data in SAC

Questions?

- Thanks for your attention!

- This work is supported by the Center for Energy and Geo Processing (CeGP) at Georgia Tech and by King Fahd University of Petroleum and Minerals (KFUPM).