Topics in Computational Economics

Lecture 1

John Stachurski

NYU 2016

Today's Lecture

- Overview of the course
  - Prereqs
  - Syllabus
  - Assessment
  - Resources
- Setting up a scientific coding platform
- Quick intro to Python and Julia
- Help installing software

Preliminary comments

You can reach me at `john.stachurski@anu.edu.au`

The course homepage / repo is

- https://github.com/jstac/quantecon_nyu_2016

All lectures will also be available from that repo

All homework to be uploaded to that repo (details later)

Public comments can be posted on the issue tracker

- https://github.com/jstac/quantecon_nyu_2016/issues

Preliminary comments

Please

- sign up to GitHub
- visit `jstac/quantecon_nyu_2016` and click "Watch"

Any news between classes will appear on the course homepage

Please bring laptop to class, at least for first 6 weeks

Prerequsites

Programming: Some minimal amount of experience

- Matlab
- C / Fortran
- etc

Python or Julia not required

Mathematics: Some upper level maths

- Linear algebra
- Basic analysis and probability

Aptitude is most important

Resources

Homepage https://github.com/jstac/quantecon_nyu_2016

Lecture site http://quant-econ.net/

Texts (recommended but not essential)

- Kendall Atkinson and Weimin Han (2009). Theoretical Numerical Analysis (3rd ed)
- Ward Cheney (2001). Analysis for Applied Mathematics
- Nancy Stokey and Robert Lucas Jr. (1989) Recursive Methods in Economic Dynamics
- John Stachurski (2009). Economic Dynamics: Theory and Computation

Syllabus

The structure of the course will be

1. Setting up
2. Programming in Python and Julia
3. Refresher on functional analysis and measure
   - Oriented towards numerical methods
4. Applications
   - Markov dynamics, dynamic programming, forward looking models, optimal savings, asset pricing, etc.

Details at https://github.com/jstac/quantecon_nyu_2016

Possibly subject to small changes

Assessment

- A class project 60%
  - Replicate research in Python or Julia or own topic
  - Ideally, combines code and theory from the course
- Homework assignments 25%
- Presentation(s) 10%
  - A library or programming topic
  - Your class project
- Participation 5%

Platform

In class we'll use a common computing platform based around Linux

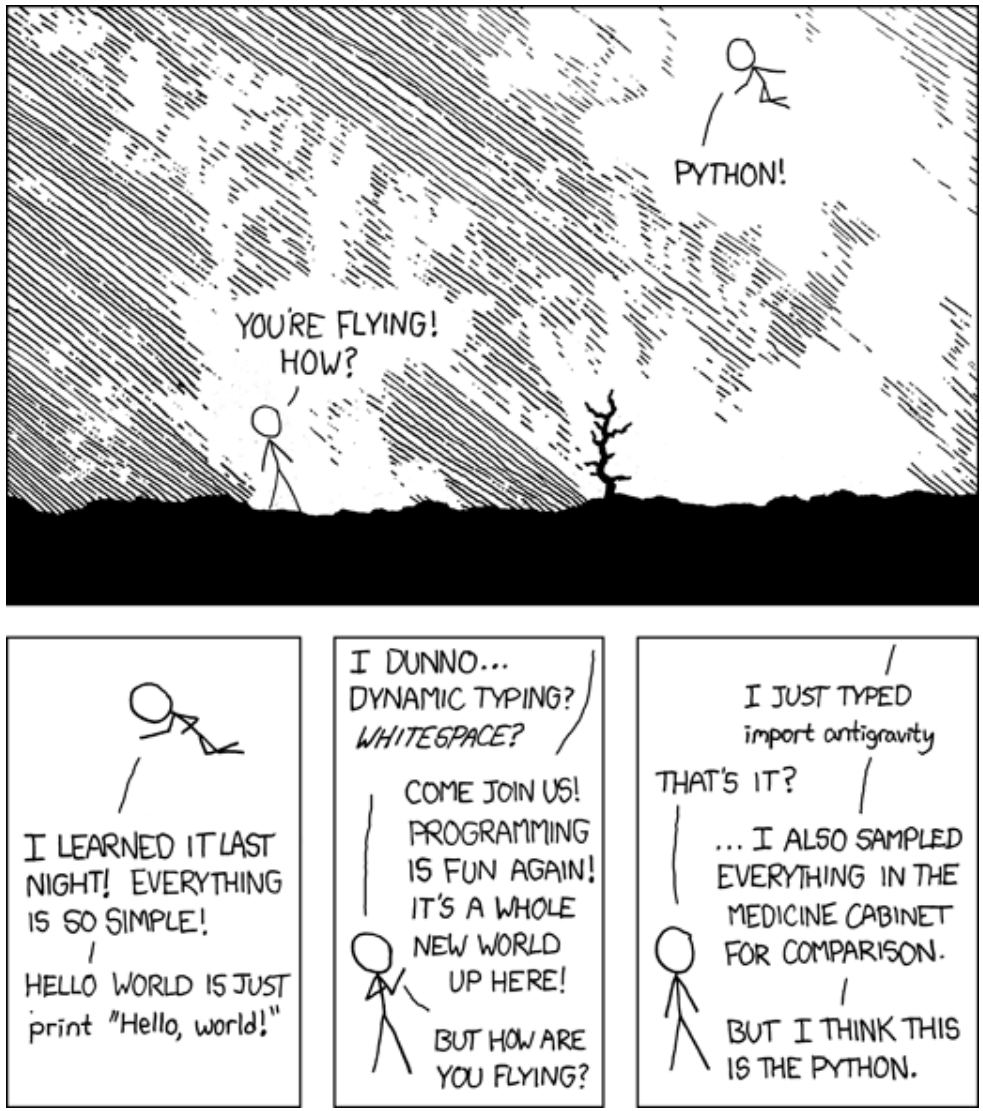
- Oriented towards power users
  - The standard desktop environ at Google
  - The standard server OS at Google
  - Emphasizes command line
- Popular choice for scientific computing

Our set up flavor will be Ubuntu but use whichever you like

On top of the platform sits the programming environment

What makes a good one?

- network effects?
- speed?
- productivity?
- fun?


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Low Level vs High Level

Low level languages:

- C, C++
- Fortran

High level languages

- Matlab
- Julia
- Python
- R, etc.

Low level languages require more details from the user

```c
/* Creates a linear array.  */
void linspace(double *ls, double a, double b, int n)
{
    double step = (b - a) / (n - 1);
    int i;
    for (i = 0; i < n; i++)
    {
        ls[i] = a;
        a += step;
    }
}
```

High level languages require less information

```
function linspace!(ls, a, b)
    n = length(ls)
    step = (b - a) / (n - 1)
    for i in 1:n
        ls[i] = a
        a += step
    end
    return ls
end
```

High level languages tend to be convenient for the user

```
>>> a, b = 1, 4
>>> a + b
5
>>> a, b = 'foo', 'bar'
>>> a + b
'foobar'
```

But harder to convert into optimized machine code

Although **big** steps forward in recent years

- Julia, Python + Numba

About Python and Julia

Both Python and Julia are **open source**

- free as in "free beer"
  - easy to share, no license hassles

"...Our license did not include the parallel processing toolbox, which means we could not do real work..."

- Source code can be freely read / modified
  - How does `pandas` computes Newey-West covariance matrices?
  - If Sphinx doesn't build your website properly then change it

Open Source Development Case Study: Julia

The code is free for the forking

- https://github.com/JuliaLang/julia

Issues are debated on the issue tracker

- https://github.com/JuliaLang/julia/issues/14670

Pull requests are accepted or rejected

- https://github.com/JuliaLang/julia/pull/14707

Open Source Development Case Study: QuantEcon.py

The code is free for the forking

- https://github.com/QuantEcon/QuantEcon.py

Issues are debated on the issue tracker

- https://github.com/QuantEcon/QuantEcon.py/issues/141

Pull requests are accepted or rejected

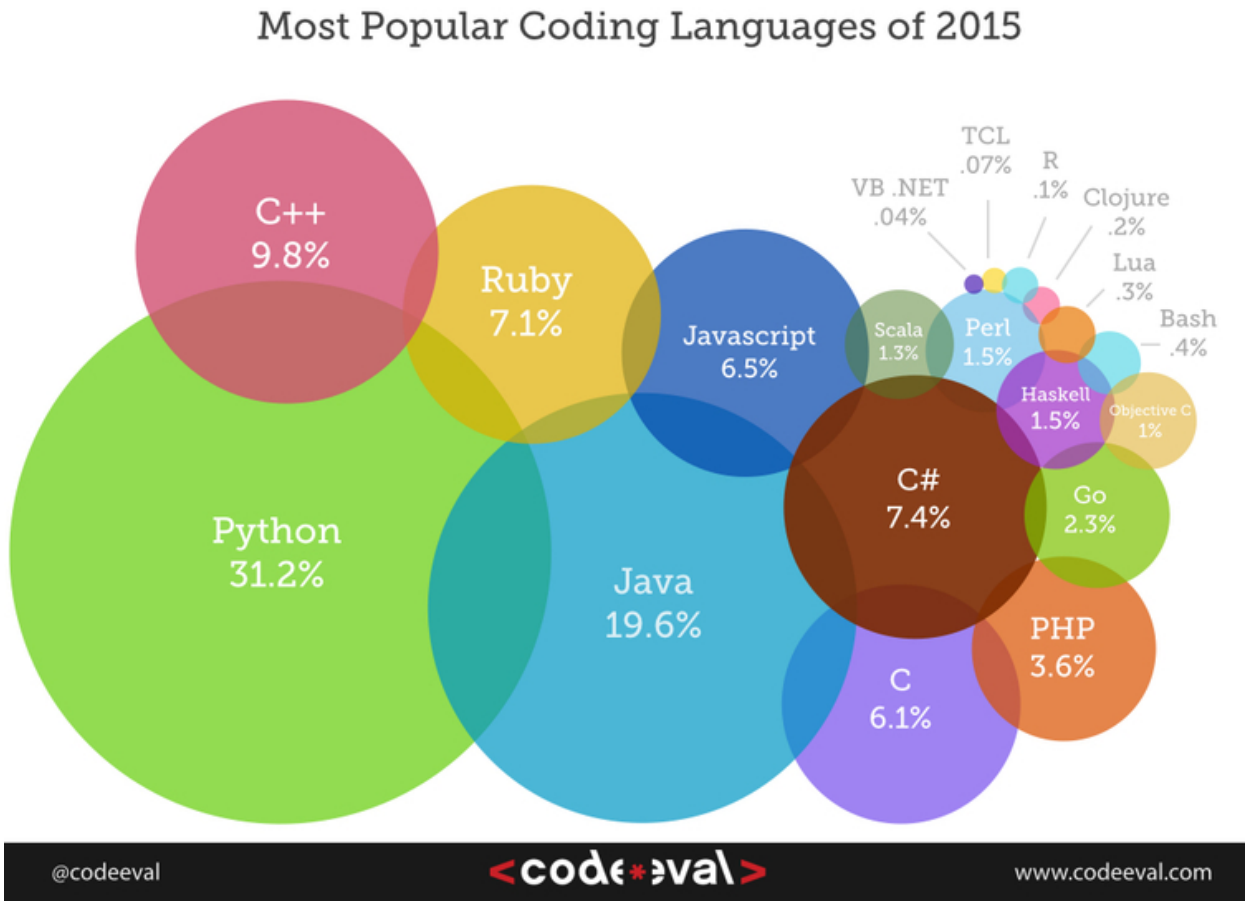
- https://github.com/QuantEcon/QuantEcon.py/pull/171

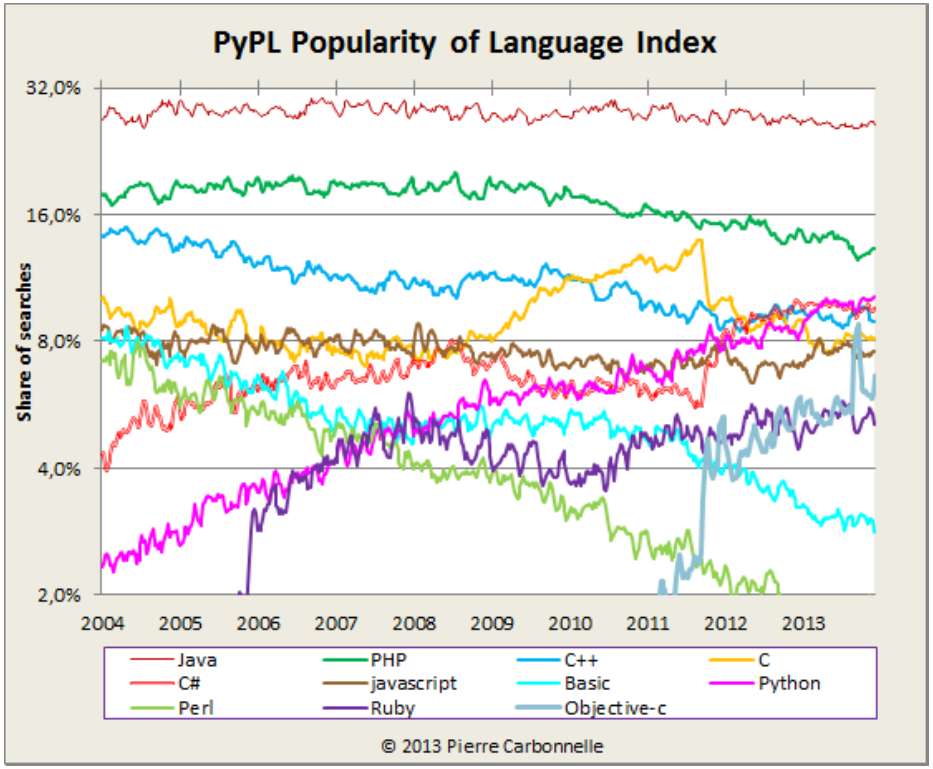
About Python

A general purpose programming language

Used extensively by

- Tech firms (YouTube, Dropbox, Instagram, Reddit, etc., etc.)
- Hedge funds and finance industry
- Gov't agencies (NASA, CERN, etc.)
- Academia


Most Popular Coding Languages of 2015
C++ 9.8%
Ruby 7.1%
Javascript 6.5%
Python 31.2%
Java 19.6%
C# 7.4%
C 6.1%
PHP 3.6%
Go 2.3%
Scala 1.3%
Perl 1.5%
Haskell 1.5%
Objective C 1%
VB .NET .04%
TCL .07%
R .1%
Clojure .2%
Lua .3%
Bash .4%
@codeeval
<code*eval>
www.codeeval.com


PyPL Popularity of Language Index
32,0%
16,0%
8,0%
4,0%
2,0%
Share of searches
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
Java
PHP
C++
C
C#
javascript
Basic
Python
Perl
Ruby
Objective-c
© 2013 Pierre Carbonnelle

Python is noted for

- Elegant, modern design
- Clean syntax, readability
- High productivity

Often used to teach first courses in comp sci and programming

- NYU Introduction to Computer Programming 2016
- Yale, MIT, Udacity, edX, etc.

```
import matplotlib.pyplot as plt

vals = []
data_file = open("ar1_series.txt"):
    for line in data_file:
        date, value = line.split(',')
        vals.append(float(value))
data_file.close()

plt.hist(vals, bins=40)
plt.show()
```

Flexibility

From enterprise level to simple system admin scripts

- https://github.com/jstac/backup_scripts

Building quant-econ.net

- Sphinx, Jinja and script files: `make html`
- Serving a local build: `python -m http.server`

Scientific Programming

Rapid adoption by the scientific community

- Artifical intelligence
- engineering
- computational biology
- chemistry
- physics, etc., etc.

Major Scientific Libraries

NumPy

- basic data types
- simple array processing operations

SciPy

- built on top of NumPy
- provides additional functionality

Matplotlib

- 2D and 3D figures

NumPy Example: Mean and standard dev of an array

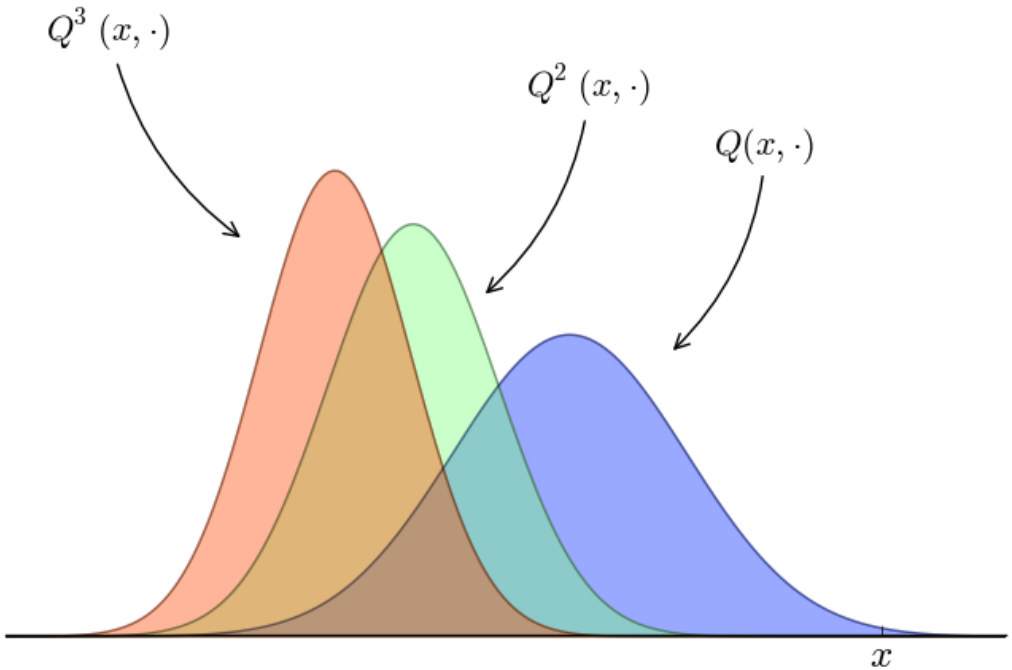
```
>>> import numpy as np
>>> a = np.random.randn(10000)
>>> a.mean()
0.0020109779347995344
>>> a.std()
1.0095758844793006
```

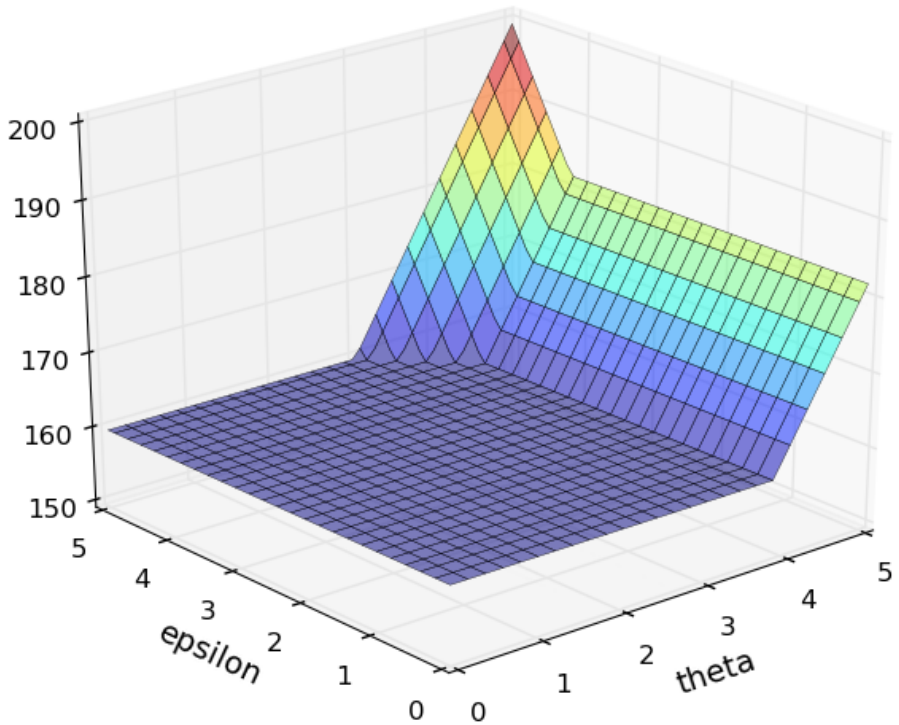
SciPy Example: Calculate

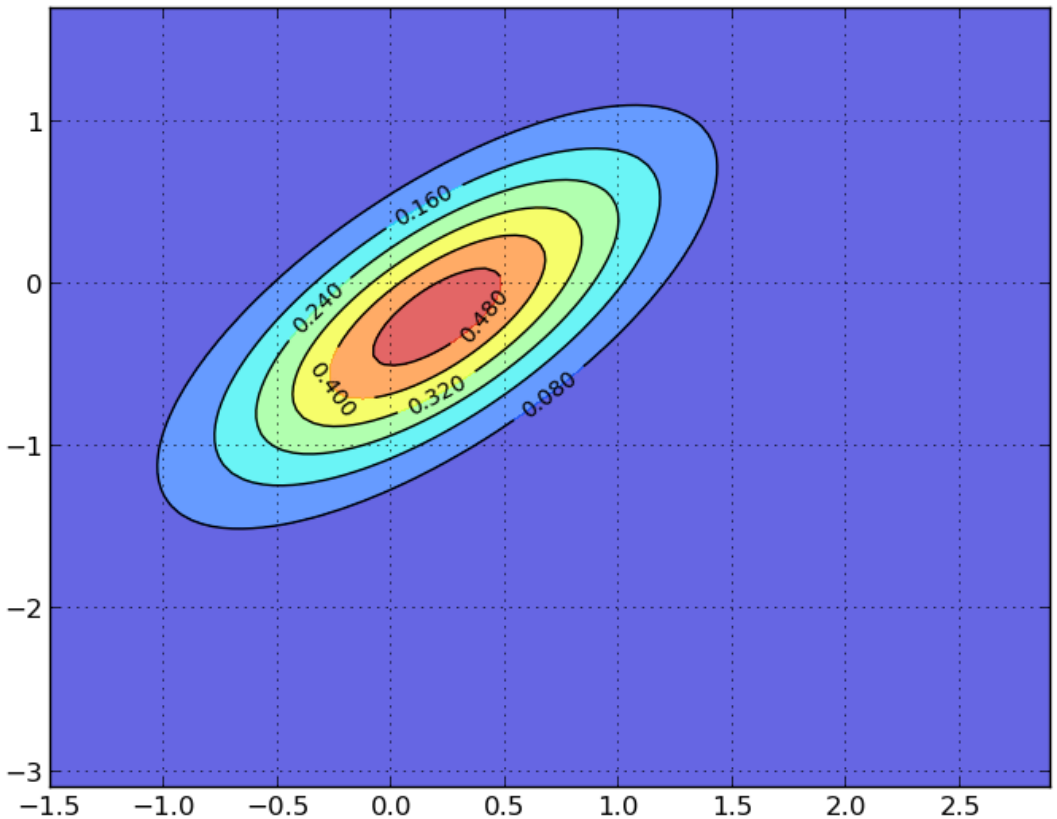
$$\int_{-2}^{2} \phi(z)dz \quad \text{where} \quad \phi \sim N(0,1)$$

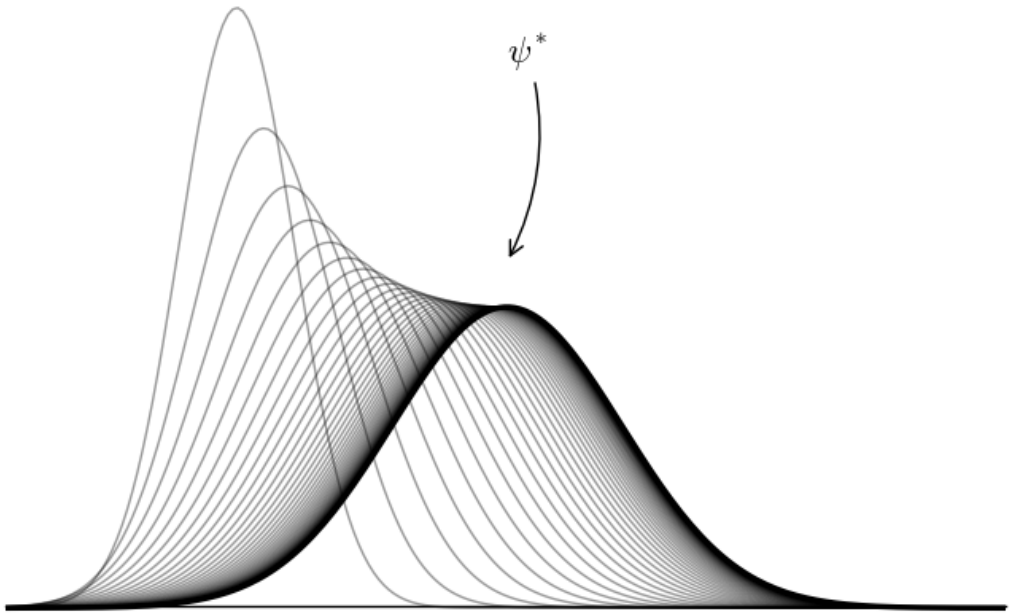
```
>>> from scipy.stats import norm
>>> from scipy.integrate import quad
>>> phi = norm()
>>> value, error = quad(phi.pdf, -2, 2)
>>> value
0.9544997361036417
```

Matplotlib examples


$Q^3\,(x,\cdot)$
$Q^2\,(x,\cdot)$
$Q(x,\cdot)$
x

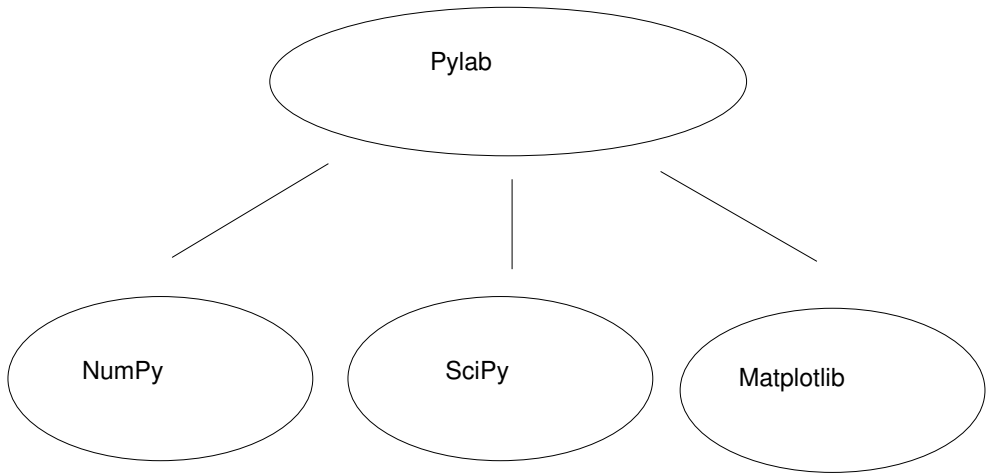

200
190
180
170
160
150
5
4
3
2
1
0
epsilon
0
1
2
3
4
5
theta


0.160
0.240
0.480
0.400
0.320
0.080
1
0
−1
−2
−3
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5


ψ^*

Pylab

Pylab combines core functionality of the big three



But now **depreciated** so please don't use

Other Scientific Libraries

Pandas

- statistics and data analysis

SymPy

- symbolic manipulations à la Mathematica

Still more:

- **statsmodels** — statistics / econometrics
- **scikit-learn** — machine learning in Python

Other Scientific Tools

Thousands of tools, for tasks like

- parallel processing, GPUs
- working with graphs / networks
- interfacing C / C++ / Fortran
- cloud computing
- etc.

For more info see http://quant-econ.net/py/about_py.html

Julia

- Recent
- Open source
- Specifically for scientific programming
- Well designed language
- Good standard library
- Growing community

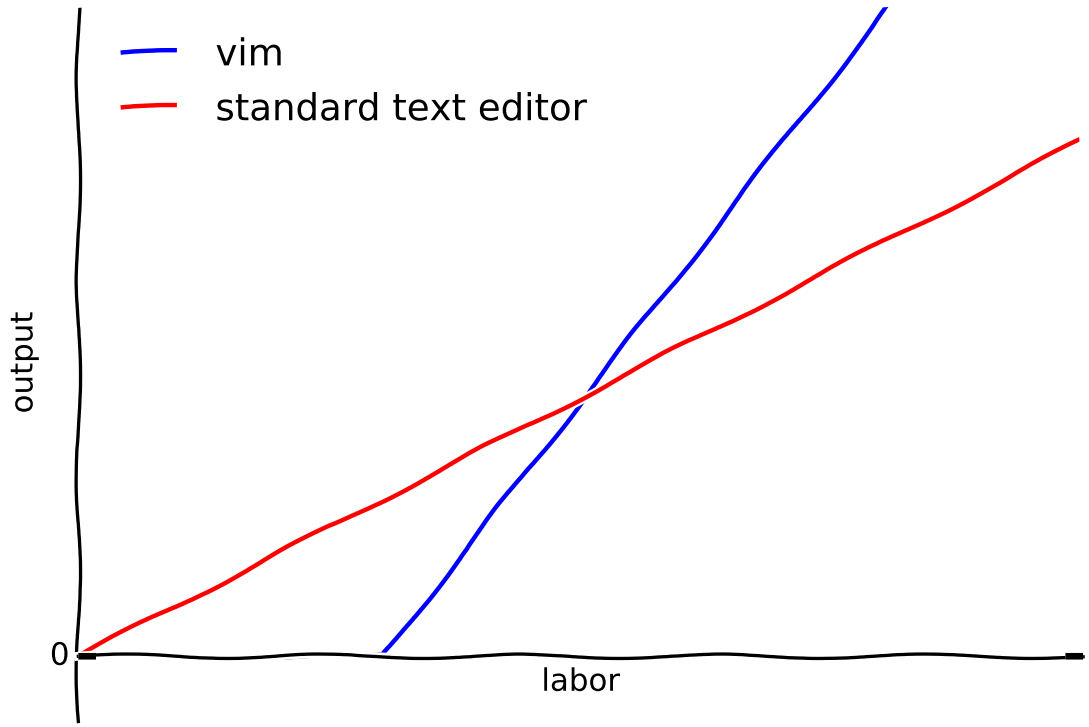
Syntax quite similar to Matlab

```
N = 500
beta = 1.0
alpha = 0.9
s = 1.0
x = zeros(N)

for i in 1:(N - 1)
    x[i+1] = beta + alpha * x[i] + s * randn()
end
```

With fast loops!

Homework: Getting Started with Vim



My perspective:

> *"Vim raises manual text editing from a mundane chore into an art that rewards study and practice, like playing guitar, swimming, or jujitsu." – someone on the Internet*

Another perspective:

> *"Don't evangelize editors because one day you'll look like an ass." – someone else on the Internet*

Undeniably correct:

> *Choose a really good text editor and use it for everything (all coding, TeX, command line, emails, etc.) – me, and lots of other people*

Your mission is to try vim for the duration of this course

- http://www.vim.org/about.php

Install

- `sudo apt-get install vim`

Daily practice

- Run `vimtutor` at terminal and practice for 20 minutes

Read:

- https://danielmiessler.com/study/vim/

Configure your `.vimrc` (which lives in `~`)

- A minimal configuration file: https://gist.github.com/jstac/3bec513653382a4a903b
- http://benmccormick.org/2014/07/14/learning-vim-in-2014-configuring-vim/

Read about Vundle and install

- https://github.com/VundleVim/Vundle.vim