SoftwareInfrastructure

Final 1.0

Brian Granger

May 31, 2013

Part I

The IPython Notebook

1 Software Infrastructure for Reproducibility

2 Brian E. Granger

I **teach** Physics and do **research** with undergraduates:

I am a **developer** of tools for scientific and technical computing:

3 Research workflows and reproducibility

The research workflow:

- Research requires a series of **tasks**
- A sequence of tasks is a **workflow**
- The goal of that workflow is to tell a **story**

- The exploratory nature of research means that the steps and workflow are ever changing
- Reproducibility enables other people to explore, verify, modify and retell that story for themselves.

These workflows span the entire **lifecycle** of research:

- Individual exploration
- Collaborative development
- Production execution
- Debugging
- Publication
- Presentation
- Education

This lifecycle, and thus reproducibility, includes publication, presentation and education. If you change a parameter in a calculation, which changes a figure, your conference talk and class lecture notes need to be updated.

4 What is our tool?

http://ipython.org

The IPython Notebook is a web-based **computing environment** and **open document format** for telling stories with code and data that are:

- Interactive
- Exploratory
- Collaborative
- Open
- Reproducible
- Productive
- Fun

5 Reproducibility demo

Zach Sailer is a Cal Poly senior physics major who has done his senior thesis on liquid crystals using the IPython Notebook. He has posted his data and notebooks on GitHub:

https://github.com/Zsailer/calpolythesis

Let's reproduce his research! Run the following at the command line:

```
git clone git://github.com/Zsailer/calpolythesis.git
cd calpolythesis/notebooks
ipython notebook
```

This will open the Notebook server and will enable you to open and run his Notebooks. Important points:

- You can tweet about it - only a single click in a browser to view it on http://nbviewer.ipython.org
- You can run it yourself in a short amount of time - download and “Run All”
- You can hack on it and explore it yourself
- It is version controlled on a public GitHub repo - completely transparent
- Most of the reproducibility is a side effect of the tools being used

6 How does a user install our tool?

1. They don’t, it is a web application so they only need a modern web browser
2. Someone has to run the IPython Notebook server:

In the cloud:

- PiCloud
- Wakari
- StarCluster

Double clickable installers:

- Continuum’s Anaconda
- Enthought Python Distribution (EPD)
- Enthought’s Canopy

Python package managers:

- pip
- easy_install

Linux package managers:

- apt
- yum

From source (https://github.com/ipython/ipython)

7 What types of research steps can our tool capture?

7.1 Code

The IPython Notebook is first and foremost an environment for writing and running code in many languages.

Python

Python code can be run interactively:

In [2]:
```
import time
for i in range(10):
    print i
    time.sleep(1)
```

```
0
1
2
3
4
5
6
7
8
9
```

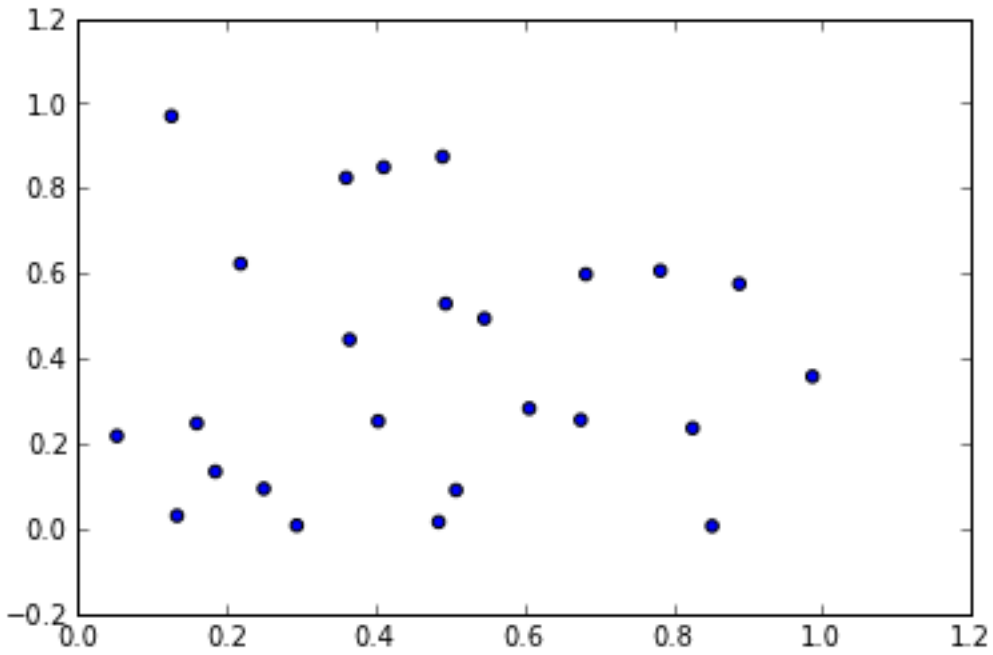
We have integrated support for plotting with Matplotlib:

In [3]:
```
%pylab inline
```

```
Welcome to pylab, a matplotlib-based Python environment [backend:
module://IPython.kernel.zmq.pylab.backend_inline].
For more information, type 'help(pylab)'.
```

In [4]:
```
scatter(rand(25),rand(25))
```

Out [4]:
```
<matplotlib.collections.PathCollection at 0x108217b50>
```



R

Using IPython's cell magic syntax (`%%R`) you can run code in other languages such as R:

```
In [5]: %load_ext rmagic
```

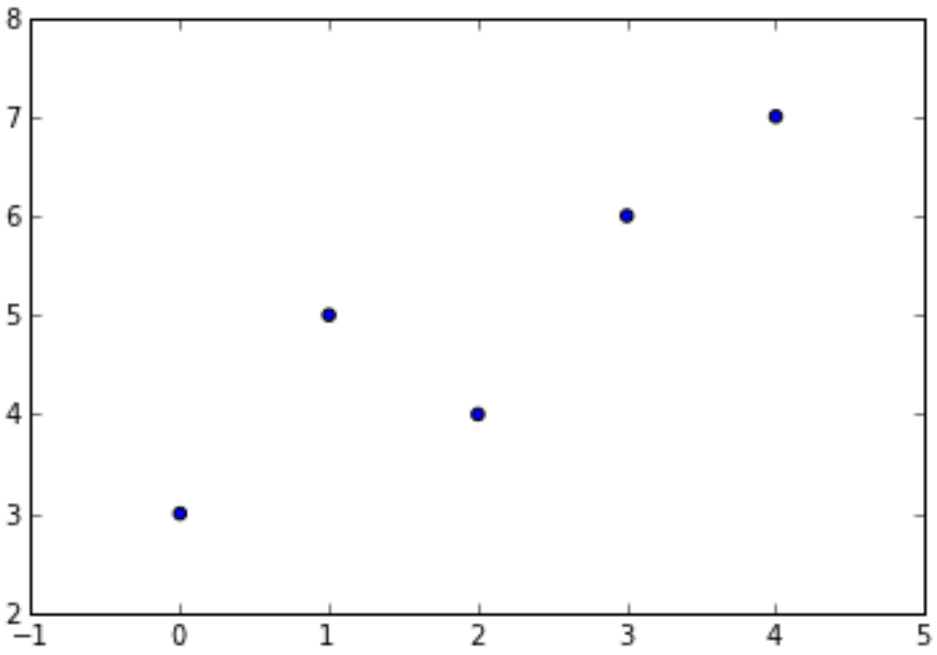
Create two Python arrays:

```
In [6]: import numpy as np
        X = np.array([0,1,2,3,4])
        Y = np.array([3,5,4,6,7])
```

Make a scatter plot in Python:

```
In [7]: scatter(X,Y)
```

```
Out [7]: <matplotlib.collections.PathCollection at 0x10cf35750>
```



Now pass those arrays to R and fit a linear model:

```
In [8]: %%R -i X,Y -o XYcoef
        XYlm = lm(Y~X)
        XYcoef = coef(XYlm)
        print(summary(XYlm))
        par(mfrow=c(2,2))
        plot(XYlm)
```

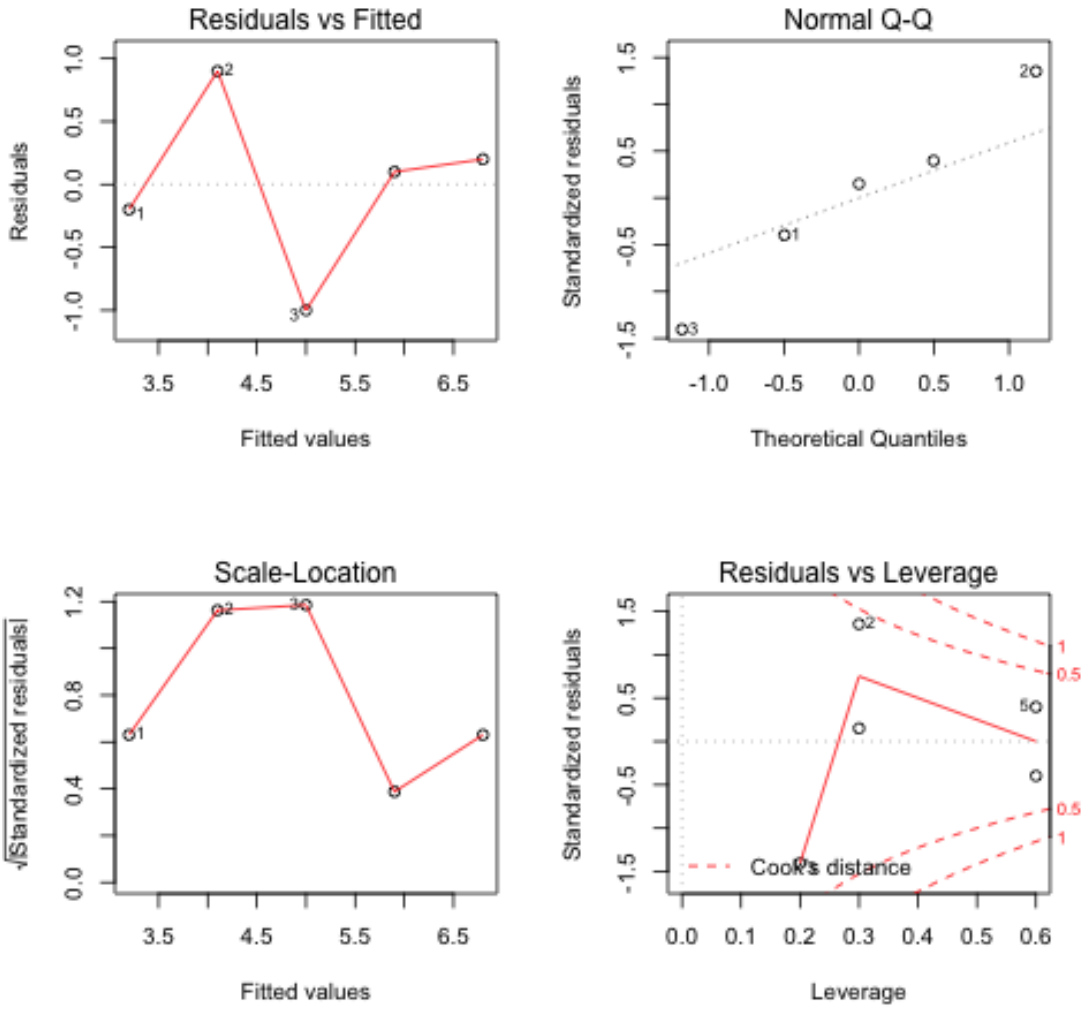
```
Call:
lm(formula = Y ~ X)

Residuals:
   1    2    3    4    5
```

```
-0.2  0.9 -1.0  0.1  0.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.2000     0.6164   5.191   0.0139 *
X             0.9000     0.2517   3.576   0.0374 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7958 on 3 degrees of freedom
Multiple R-squared:   0.81,     Adjusted R-squared:  0.7467
F-statistic: 12.79 on 1 and 3 DF,  p-value: 0.03739
```



Ruby

In [9]:
```
%%ruby
puts "Hello from Ruby #{RUBY_VERSION}"
```

```
Hello from Ruby 1.8.7
```

Bash

In [10]:
```
%%bash
echo "hello from $BASH"
```

```
hello from /bin/bash
```

Other languages

The IPython notebook supports other languages through two routes:

- Cell magics using the `%%language` syntax
- Native kernels that speak the IPython message protocol

Incomplete and growing list:

- Perl
- Cython
- Octave
- MATLAB
- node.js
- Julia
- JavaScript

7.2 Descriptive text + Equations + HTML

This is a block of descriptive text. It uses the popular Markdown syntax that is documented here. We have extended Markdown to understand LaTeX equations (using MathJax). If you are a physicist you might want to show Maxwell's equations:

$$\nabla \times \vec{\mathbf{B}} - \frac{1}{c}\frac{\partial \vec{\mathbf{E}}}{\partial t} = \frac{4\pi}{c}\vec{\mathbf{j}}$$
$$\nabla \cdot \vec{\mathbf{E}} = 4\pi\rho$$
$$\nabla \times \vec{\mathbf{E}} + \frac{1}{c}\frac{\partial \vec{\mathbf{B}}}{\partial t} = \vec{\mathbf{0}}$$
$$\nabla \cdot \vec{\mathbf{B}} = 0$$

Or possibly some inline math, such as Newton's 2nd law, $\vec{F} = m\vec{a}$. Markdown is a superset of HTML so you can also include arbitrary HTML in your text cells. This includes tables:

And images from the internet or local file system:

Header 1	Header 2
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

7.3 Output

When you run code, it can produce output. In IPython, the notion of output is very general and includes:

- Text
- Equations
- Images
- Videos
- Figures
- JavaScript widgets
- HTML

Objects can declare special logic that is used to render and display these different representations.

Equations

IPython has a `Math` object whose representation is a LaTeX formula. The Notebook detects that representation and renders it using MathJax:

In [11]:
```
from IPython.display import Math
Math(r'F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx')
```

Out [11]:

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx$$

This enables libraries, such as SymPy, to provide nice LaTeX rendering in the Notebook of live computational objects:

In [12]:
```
%load_ext sympy.interactive.ipythonprinting
```

In [13]:
```
from sympy import *
x = symbols("x")
(1/cos(x)).series(x, 0, 10)
```

Out [13]:

$$1 + \frac{1}{2}x^2 + \frac{5}{24}x^4 + \frac{61}{720}x^6 + \frac{277}{8064}x^8 + \mathcal{O}\left(x^{10}\right)$$

Images

Objects can declare image representations that are rendered by the Notebook:

In []:
```
from IPython.display import Image
Image('figures/logo.png')
```

HTML

Because the Notebook is a web application, we have access to the full power of HTML. HTML representations of objets are rendered in the Notebook by injecting the HTML into the DOM of the output area:

In []:
```
from IPython.display import HTML, display
```

In []:
```
s = """<table>
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>"""
```

In []:
```
h = HTML(s); display(h)
```

This enables third party libraries, such as Pandas, to provide nicely formatted HTML output of their objects:

In []:
```
import pandas
pandas.core.format.set_printoptions(notebook_repr_html=True)
```

In []:
```
%%file data.csv
Date,Open,High,Low,Close,Volume,Adj Close
2012-06-01,569.16,590.00,548.50,584.00,14077000,581.50
2012-05-01,584.90,596.76,522.18,577.73,18827900,575.26
2012-04-02,601.83,644.00,555.00,583.98,28759100,581.48
2012-03-01,548.17,621.45,516.22,599.55,26486000,596.99
2012-02-01,458.41,547.61,453.98,542.44,22001000,540.12
2012-01-03,409.40,458.24,409.00,456.48,12949100,454.53
```

In []:
```
df = pandas.read_csv('data.csv'); df
```

The possibilities are endless and our users are doing amazing things with these capabilities. Here we embed a YouTube video into the Notebook:

In [14]:
```
from IPython.display import YouTubeVideo
YouTubeVideo('sjfsUzECqK0')
```

Out [14]:

JavaScript

The JavaScript representation allows you to build objects that have interactivity in the browser:

In [15]:
```
from IPython.display import Javascript
```

In [16]:
```
js="""
var rme = $('<button>Reproduce Me!</button>').
    button().click(function () {
        alert('Reproducing research...');
    });
$('#maintoolbar').append(rme);
"""
Javascript(js)
```

Out [16]: <IPython.core.display.Javascript at 0x1133b5a10>

8 How does our tool capture these steps?

- A user types code and text into **cells**
- Running the code produces **output**
- We capture **everything** (code, text and output) automatically in a **notebook document**
- Reproducibility is a side effect of simply using the tool
- Notebook documents are stored as JSON files on your local file system

- Notebook documents can be converted to static HTML, Markdown, blog posts, LaTeX/PDF, slideshows, etc.

9 How are the steps represented?

- Individual steps are represented as code or text cells
- The overall workflow is represented as a linear sequence of cells

10 How portable are the workflows?

10.1 View

To **view** a workflow, researchers can post their Notebook on any public website (GitHub!). Then, anyone else can view a static HTML rendering of the Notebook at:

http://nbviewer.ipython.org

This portability only requires a modern web browser.

10.2 Run

To **run** a workflow, users can download a Notebook and all of its assets and run it on their local system. The downloaded Notebook is live runnable code that can be changed easily and re-run for different parameters, models, datasets, etc. This requires:

- Python
- IPython and its dependencies (PyZMQ, tornado, jinja2, etc.)
- The Notebook and its related assets (data, libraries)
- Any additional dependencies the notebook has (GPUs, multicore, cluster, cloud, databases, etc.)

The Notebook runs on all of the major operating systems, and all recent versions of Python. You can run the IPython Notebook in the cloud or on a supercomputer.

11 How does the tool support sharing and collaboration?

Sharing and collaboration are important aspects of reproducibility. How does this relate to the Notebook?

- Distributed version control systems, in particular Git/GitHub, have transformed the sharing of and collaboration on code. Let's not reinvent that!
- We have gone to great lengths to make sure that the Notebook document format is version control friendly. If you change a single line of code in the Notebook, it will lead to a single line diff.
- This opens up the full collaboration model of Git/GitHub for Notebooks. If you post your Notebooks on GitHub, I can fork/clone your repo and submit pull requests!
- Researchers are sharing their Notebooks on GitHub and providing links to the rendered versions at http://nbviewer.ipython.org/
- GitHub and nbviewer URLs are emerging as a natural way to share and link to computational research.

12 Does your tool provide support for archival and longevity?

We have designed the Notebook document format with longevity in mind:

- It is an open format with a test suite
- The format is versioned to allow us to evolve the format over time while providing forward compatibility. That Notebook you wrote 10 years ago will always be readible.
- The format is based on open data formats: JSON, JPEG, PNG, HTML, etc.
- It is a simple, non-binary format. This means it is trivial to read, write, modify Notebook document programmatically from any language

The Notebook does not explicitly address archival questions. But version control systems are great for this!

Part II

Other information

13 The IPython project

- IPython: open source (BSD) interactive computing environment in Python
- History:
  - Started in 2001 by Fernando Perez (BDFL)
  - The project now has an extremely dedicated and talented team
- $>$ 20 person years of development, $>$ 150 contributors
- IPython is the de facto standard environment for interactive work in Python
- Funded by:
  - Mostly by volunteers
  - NASA, DOD/DRC, NIH
  - Microsoft, Enthought
  - Alfred P. Sloan Foundation ($1.15 million dollar grant starting in Jan. 2013)
- Components:
  - IPython Kernel
    - Stateful computation engine
    - Runs code and returns results

        * Uses language agnostic JSON based message protocol over ZeroMQ/WebSockets
    - Frontends:
        * Terminal Console
        * Qt Console
        * Notebook
    - Parallel computing framework

14 Example Notebooks

Here are the official IPython Demo Notebooks that go into more detail about the capabilities of the Notebook.

We have collected highlights of user's work in our notebook gallery

15 The future

- Notebook conversion (2012)
    - To and from a wide range of formats
    - HTML, LaTeX, PDF, Markdown, reStructured Text
- Interactive widgets (2012)
    - The IPython kernel already knows how to send JSON representations of objects to the browser
    - We are developing an architecture that will allow interactive JavaScript widgets and Python objects to pass JSON data back and forth.
    - Imagine all of your d3 visualizations backed by the power of Python/NumPy/SciPy/Pandas/etc.
    - Automatic generation of UI controls
- Multiuser Notebook server (2013)
    - Small to medium groups of users
    - Trusted users - you would give these folks shell accounts

See our development roadmap for details