# The State of the Stack

Jake VanderPlas @jakevdp
SciPy Keynote; July 10, 2015

# The State of the Stack

Jake VanderPlas @jakevdp

SciPy Keynote; July 10, 2015

Gordon and Betty MOORE FOUNDATION
ALFRED P. SLOAN FOUNDATION
Washington Research FOUNDATION
NSF

# This Talk . . .

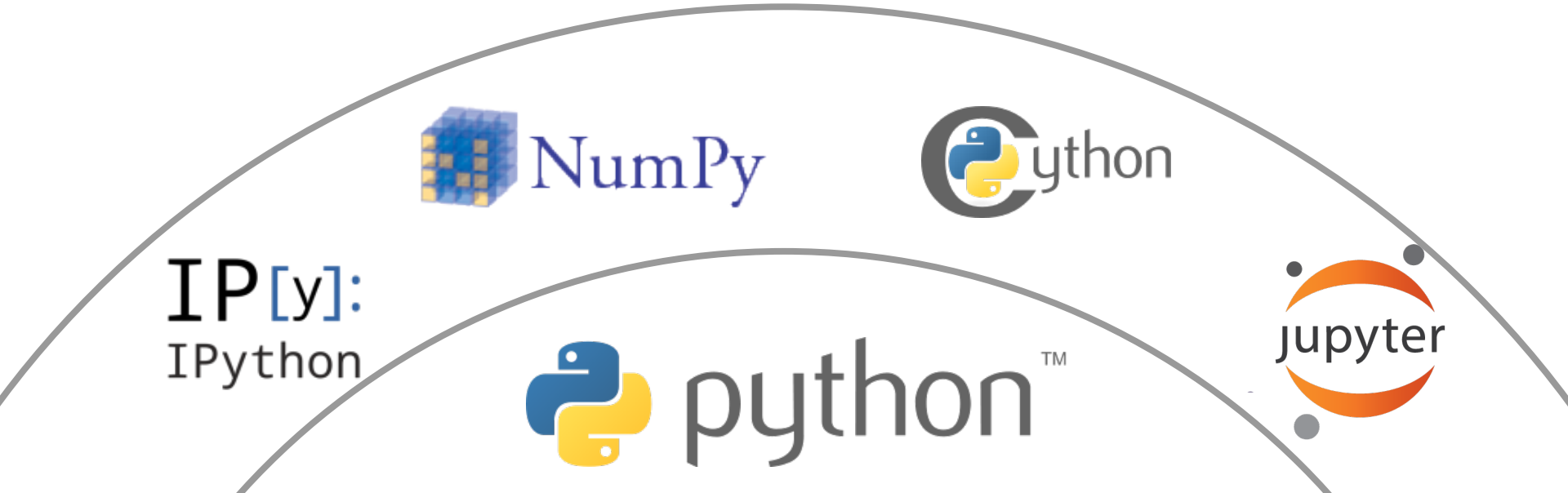
Where we are

Where we've come from

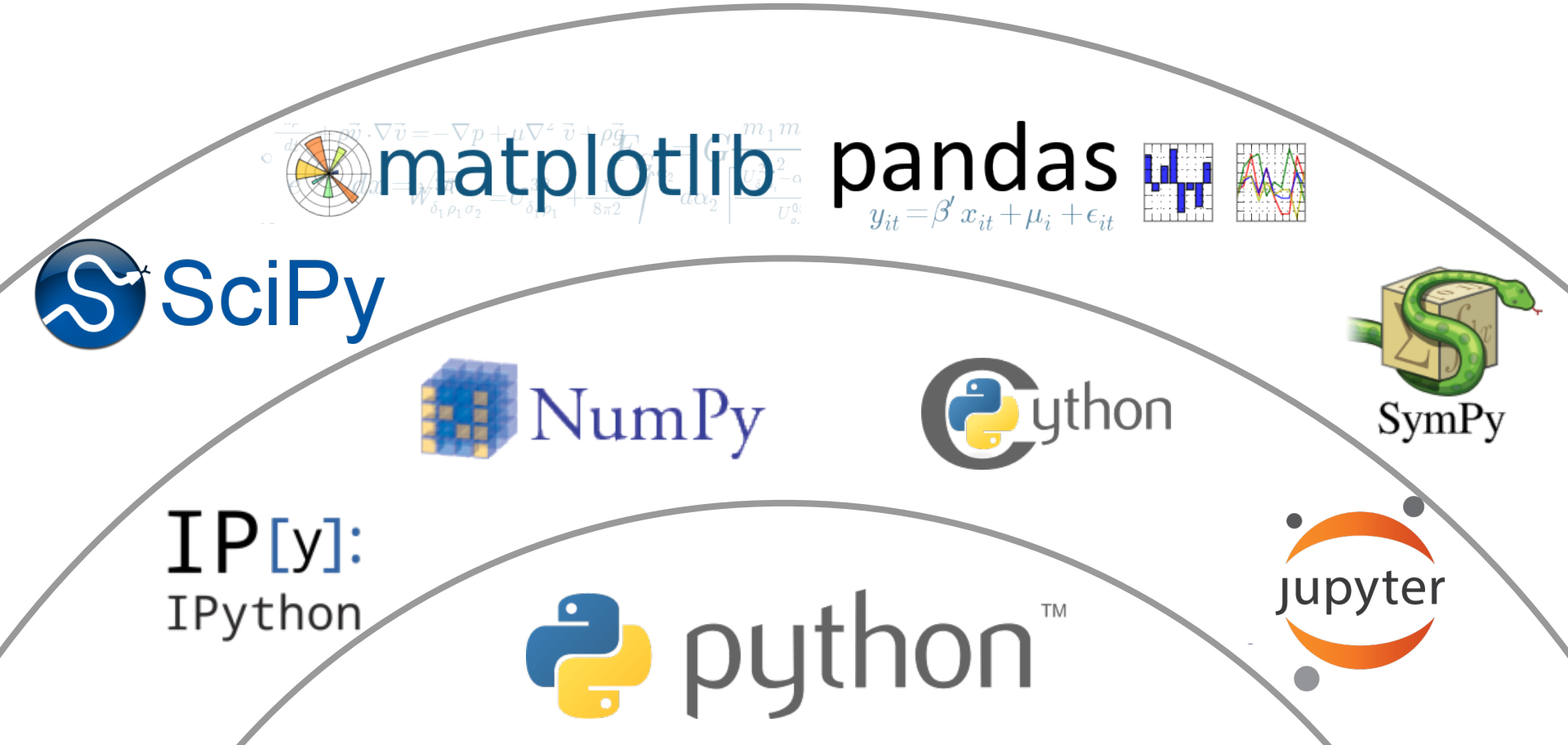
Where we're going

# Python's Scientific Ecosystem

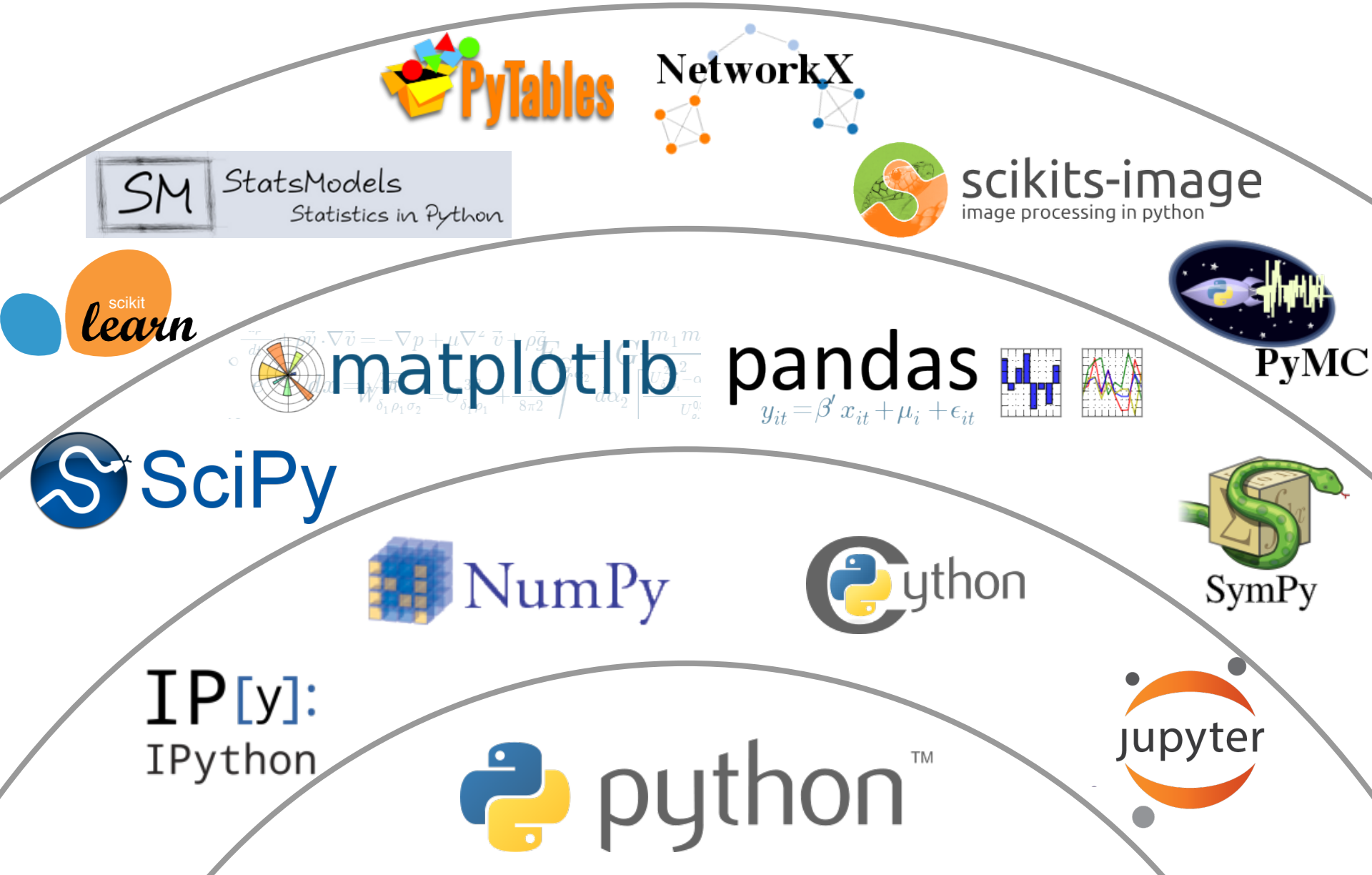
# Python's Scientific Ecosystem



# Python's Scientific Ecosystem



# Python's Scientific Ecosystem

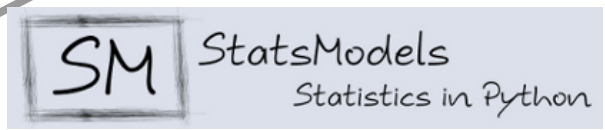






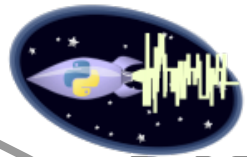

(and many, many more)
















# Many more tools:

**Performance:**

Numba, Weave, Numexpr, Theano . . .

**Visualization:**

Bokeh, Seaborn, Plotly, Chaco, mpld3, ggplot, MayaVi, vincent, toyplot, HoloViews . . .

**Data Structures & Computation:**

Blaze, Dask, DistArray, XRay,

Graphlab, SciDBpy, pySpark . . .

**Packaging & distribution:**

pip/wheels, conda, EPD, Canopy, Anaconda ...

# Recent-ish Developments

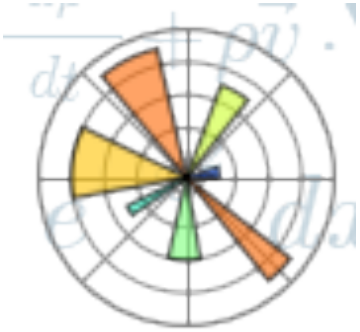
# Recent Developments: Core Language

I'll just leave this right here . . .


If you haven't switched, it's time.

# Recent Developments: Visualization

# Matplotlib:
# Evolving into a more Modern Package



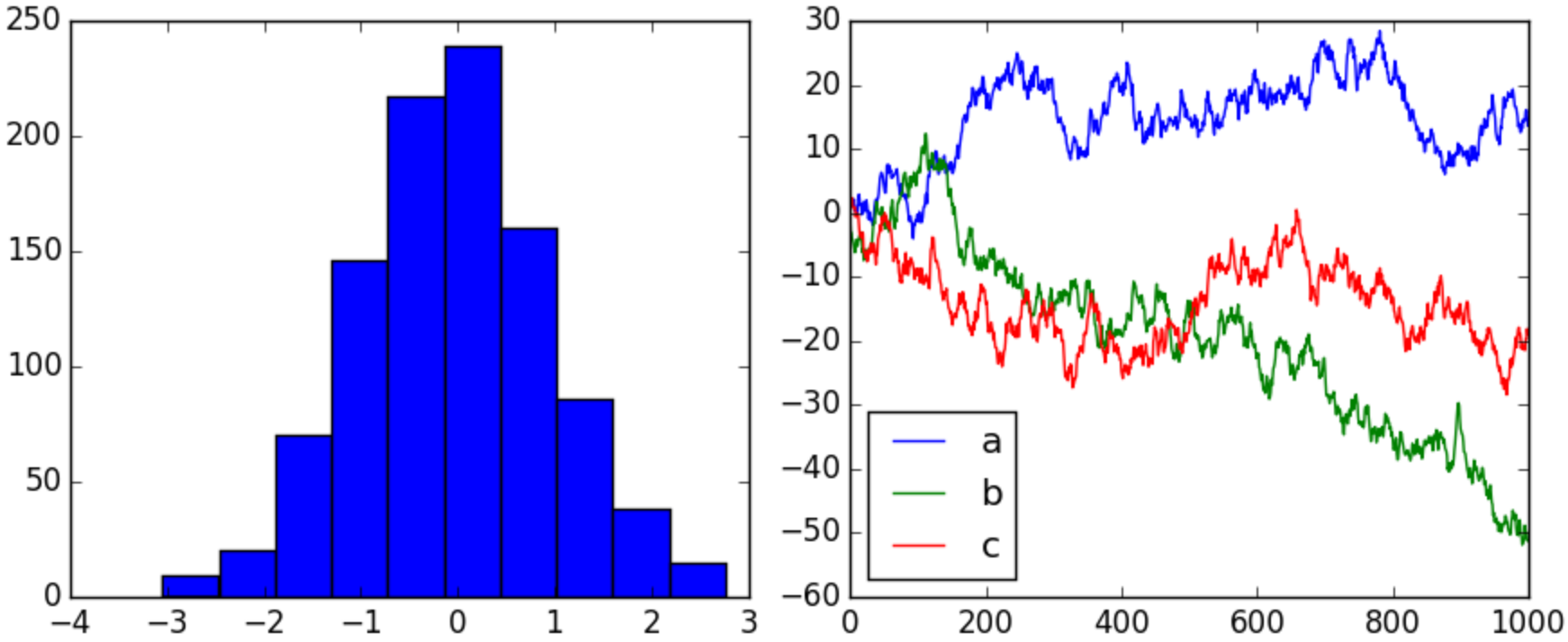
Matplotlib 1.4 features *stylesheets*,
with several very nice built-in styles.

Matplotlib 2.0 will break backward compatibility
to provide *new plot style defaults!*

(See *State of Matplotlib* talk for more details)

# New Matplotlib Styles:

In [3]:
```
make_plots()
```

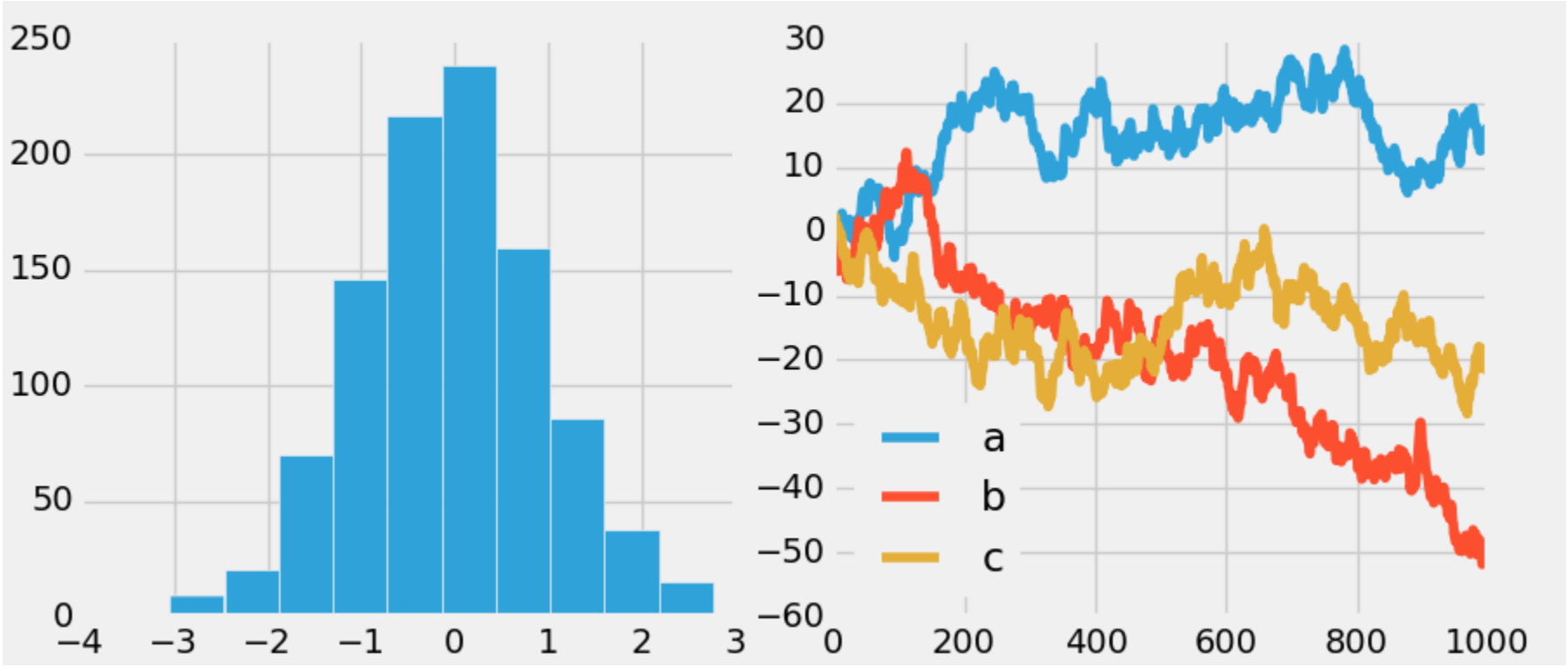


# New Matplotlib Styles:

In [4]:
```
plt.style.use('ggplot')
make_plots()
```

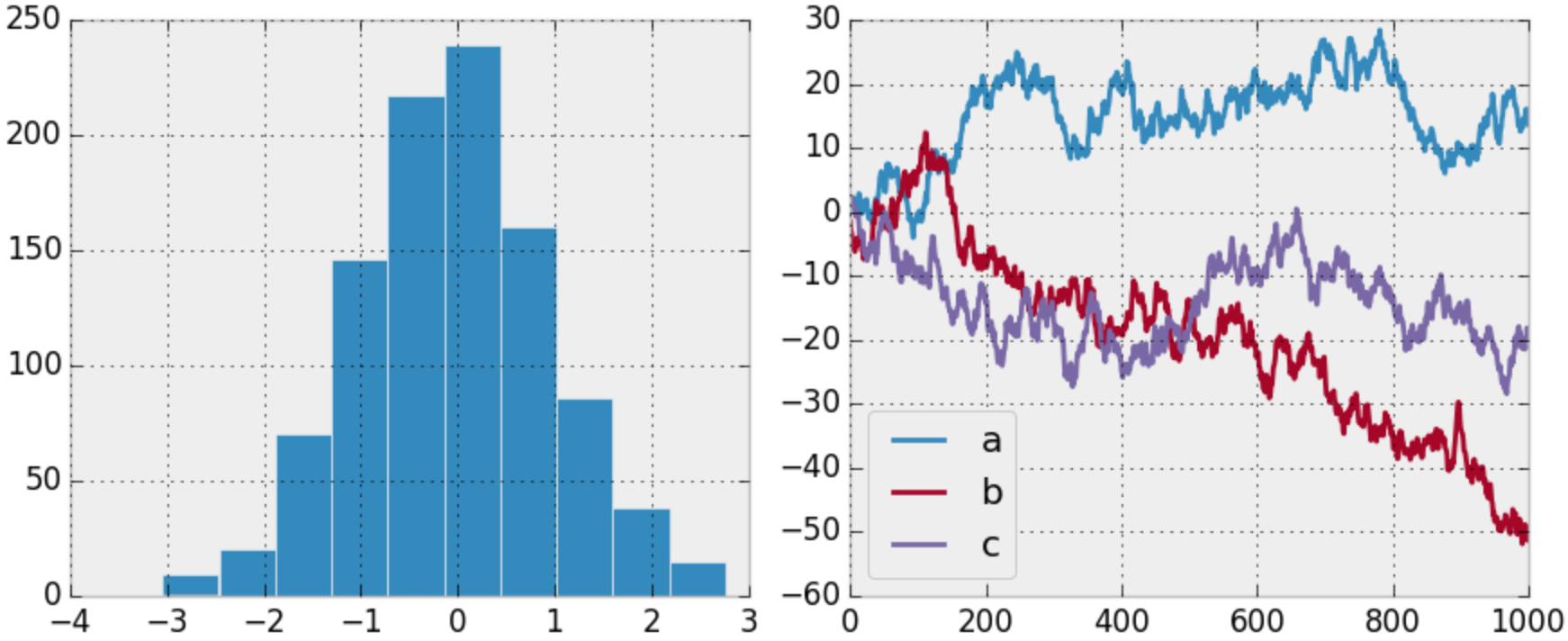
# New Matplotlib Styles:

In [5]:
```
plt.style.use('fivethirtyeight')
make_plots()
```

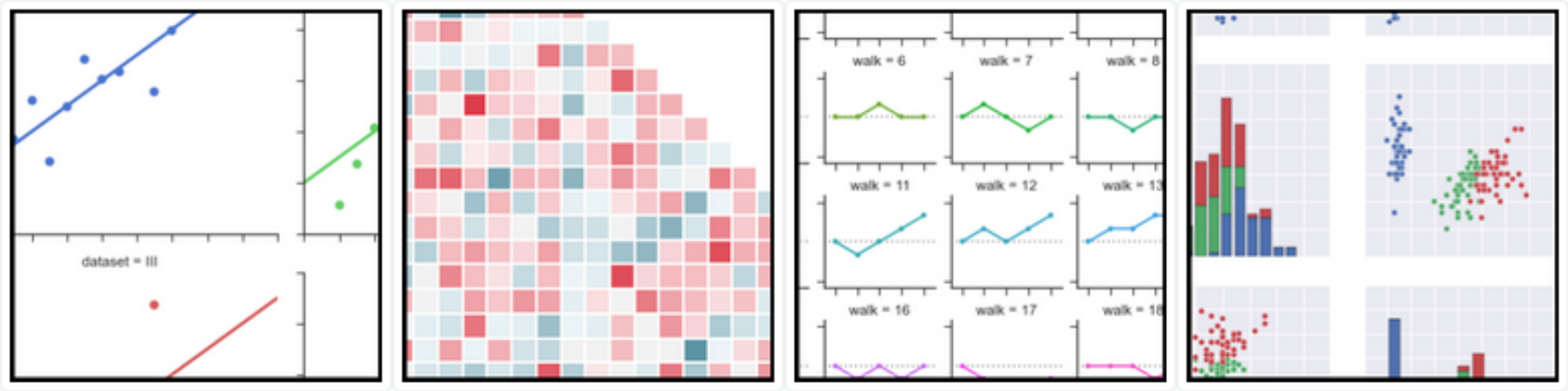


# New Matplotlib Styles:

In [6]:
```
plt.style.use('bmh')
make_plots()
```



# Seaborn: Matplotlib + Pandas + Statistical Visualization

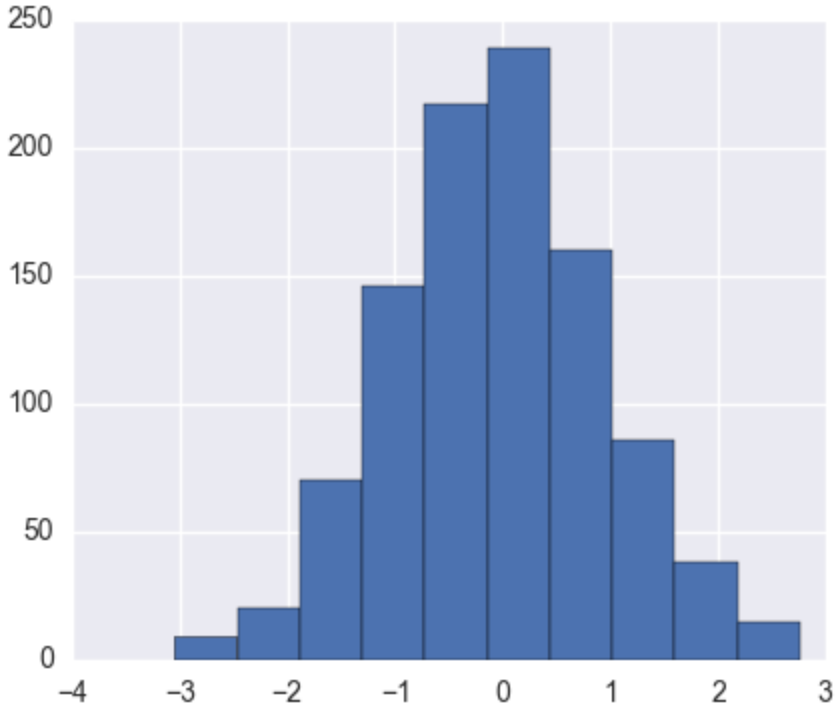


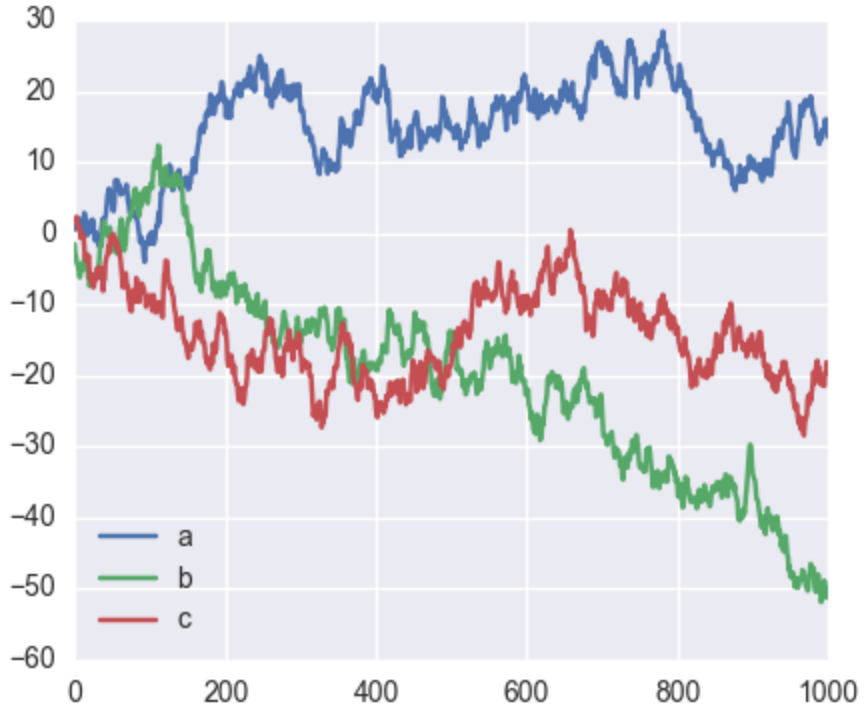

- built on top of **matplotlib**: able to use any of its backends & output formats
- **pandas**-aware: quick plotting of labeled data
- provides beautiful, well-thought-out default plot styles

http://stanford.edu/~mwaskom/software/seaborn/

# Seaborn's Matplotlib Style:

```
In [7]: import seaborn; seaborn.set()
make_plots()
```




(style available natively in next matplotlib release)

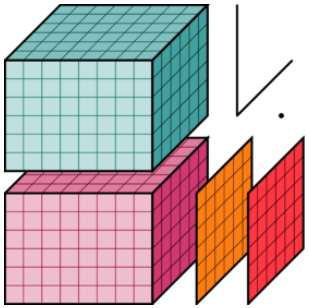
# Bokeh: Powerful Interactive Viz


- HTML5 output, both server and client-side
- Flexible in-browser interactivity
- Fundamentally a **Javascript library** with Python bindings

http://bokeh.pydata.org/

# Recent Developments: Arrays & Data Structures

# Arrays and Data Structures



**xray**

`xray` implements numpy-style ND arrays with Pandas-style labels & indices.

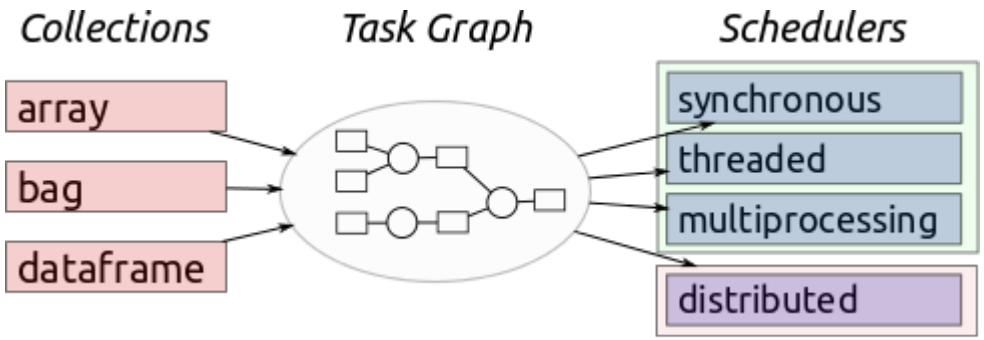
( NumPy + pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$ )

Modern data is heterogeneous, noisy, and complicated. Anonymous dense arrays are no longer enough!

http://xray.readthedocs.org/

# Arrays and Data Structures

**Dask:** a lightweight tool for general parallelized array storage and computation.




The project is still young, but the possibilities are very exciting!

http://dask.pydata.org/

# Recent Developments: Computation & Performance

# Computation & Performance:

**Numba:** with a simple decorator, Python JIT compiles to LLVM and executes at near C/Fortran speed!

```
def fib(n):
    a, b = 0, 1
    for i in range(n):
        a, b = b, a + b
    return a

%timeit fib(50)
```

```
100000 loops, best of 3: 3.83 µs per loop
```

Still some features missing, but very promising (see my blog posts for some examples).

http://numba.pydata.org/

# Computation & Performance:

**Numba:** with a simple decorator, Python JIT compiles to LLVM and executes at near C/Fortran speed!

```
@numba.jit
def fib(n):
    a, b = 0, 1
    for i in range(n):
        a, b = b, a + b
    return a

%timeit(fib(50))
```

```
1 loops, best of 3: 468 ns per loop
```

20x speedup!

Still some features missing, but very promising (see my blog posts for some examples).

http://numba.pydata.org/

# Recent Developments: Distribution & Packaging

# Distribution & Packaging:


`conda` distribution & packaging tool has changed the way many use, develop, & teach Python.

- like **pip**, but better management of Python & non-python dependencies
- like **virtualenv**, but allows different versions of compiled libraries
- Similar to **yum / apt / macports / brew**, but platform-independent!

http://conda.pydata.org/

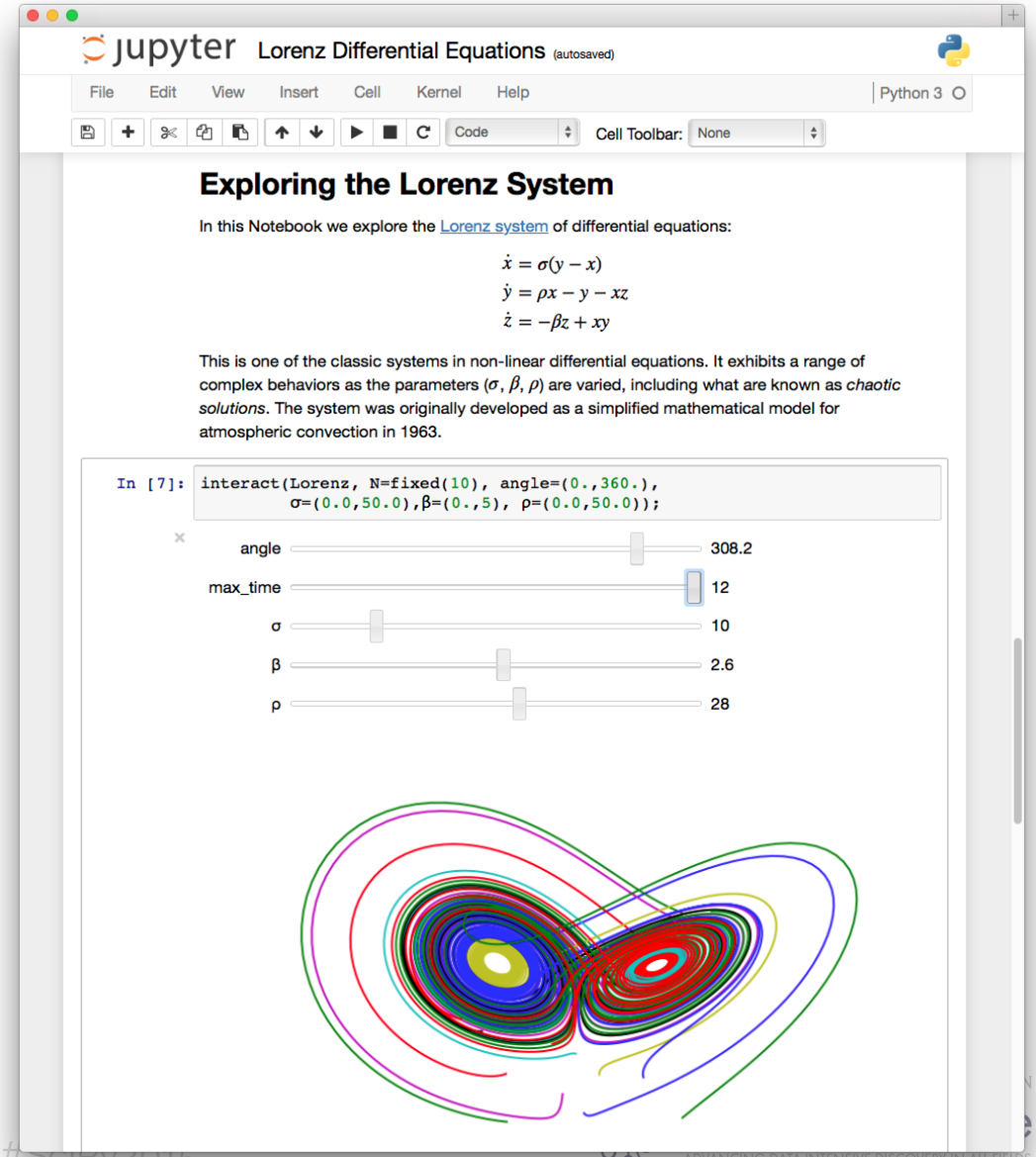
# And of course . . .

# IPython & Jupyter

So much happening . . .

- The IPython/Jupyter split
- Widgets = awesome
- Docker-based backends
- Jupyter Hub
- new $6M grant this week!

Python stack is branching out to benefit other languages!



# And so much more . . .

# A Historical Perspective . . .

# Why Python?


Python was created in the 1980s as a teaching language, and to "bridge the gap between the shell and C" [1]

1. Guido Van Rossum The Making of Python

"I thought we'd write small Python programs, maybe 10 lines, maybe 50, maybe 500 lines — that would be a big one"

Guido Van Rossum The Making of Python

**Python is *not* a scientific programming language!**

. . . why did a "toy language" become the core of a scientific stack?

# Pre-Python workflows:

"I had a hodge-podge of work processes. I would have Perl scripts that called C++ numerical routines that would dump data files, and I would load them up into MatLab to plot them. After a while I got tired of the MatLab dependency… so I started loading them up in GnuPlot."

-**John Hunter**
creator of Matplotlib
*SciPy 2012 Keynote*

# Pre-Python workflows:

"My advisor had a heavily customized awk/sed/bash workflow to manage job submissions and postprocessing of C codes for supercomputing runs… So I used her scripts to run my jobs, and on top of that had added my own layer of Perl, plus a hefty amount of Gnuplot, IDL and Mathematica."

- **Fernando Perez**
creator of IPython
*via email*

# Pre-Python workflows:

"Prior to Python, I used Perl (for a year) and then Matlab and shell scripts & Fortran & C/C++ libraries. When I discovered Python, I really liked the language... But, it was very nascent and lacked a lot of libraries. I felt like I could add value to the world by connecting low-level libraries to high-level usage in Python."

- **Travis Oliphant**
  creator of NumPy & SciPy
  *via email*

# Python is *not* a scientific programming language!

**Python is *not* a scientific programming language!**

**. . . Python is a *glue*.**

Python glues together this hodge-podge of scientific tools.

Python glues together this hodge-podge of scientific tools.

High-level syntax wraps low-level C/Fortran libraries, which is (mostly) where the computation happens.

Python glues together this hodge-podge of scientific tools.

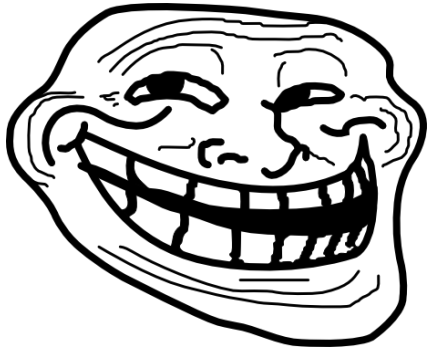
High-level syntax wraps low-level C/Fortran libraries, which is (mostly) where the computation happens.

It is ***speed of development***, not necessarily ***speed of execution***, that has driven Python's popularity.

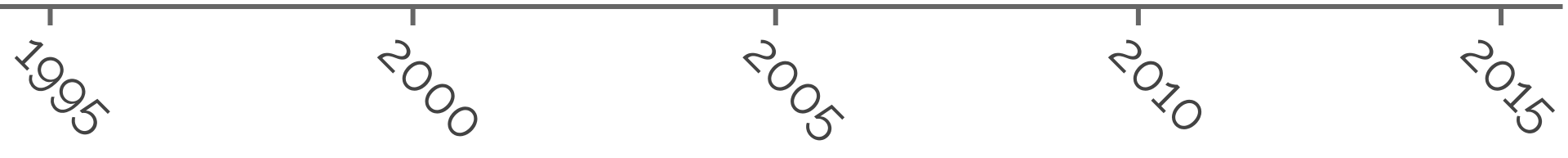
Why don't you use C instead of Python? It's so much faster!

Why don't you use C instead of Python? It's so much faster!

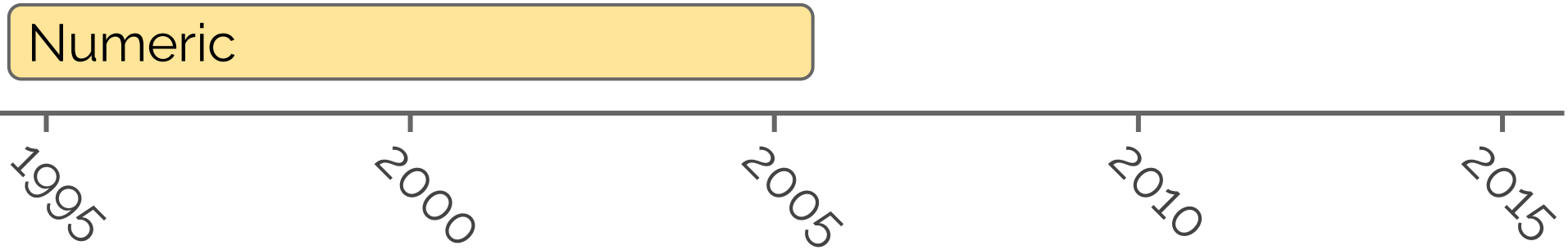
Why don't you commute by airplane instead of by car? It's so much faster!



# But this efficiency depends on the Scientific Stack . . .



1995: **Numeric** was an early Python scientific array library, largely written by Jim Hugunin



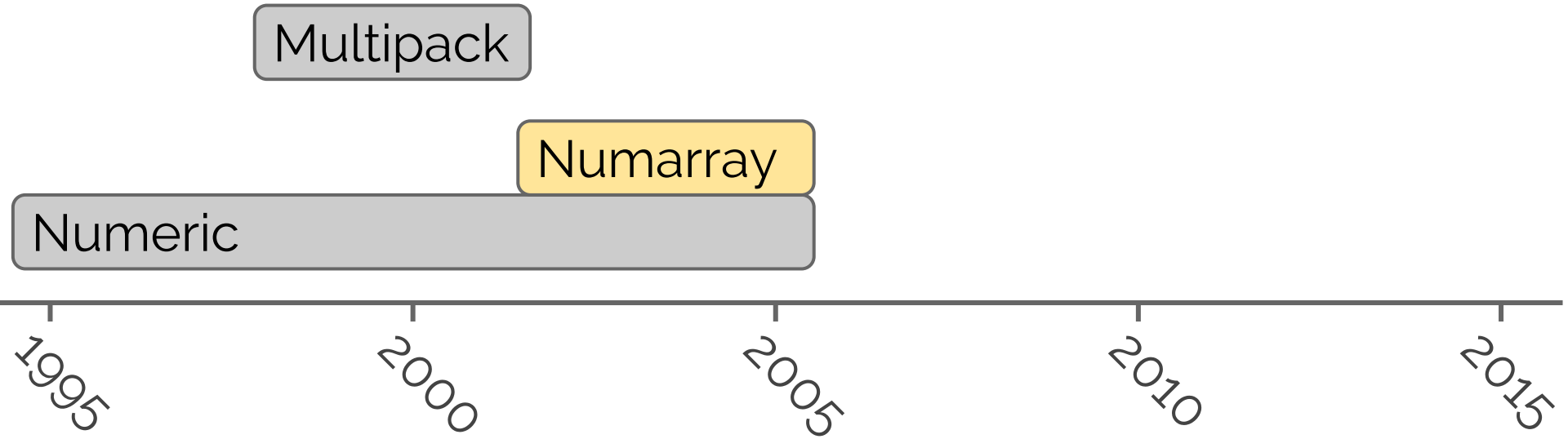
1998: **Multipack**, built on Numeric, was a set of wrappers of Fortran packages written by Travis Oliphant.

Multipack

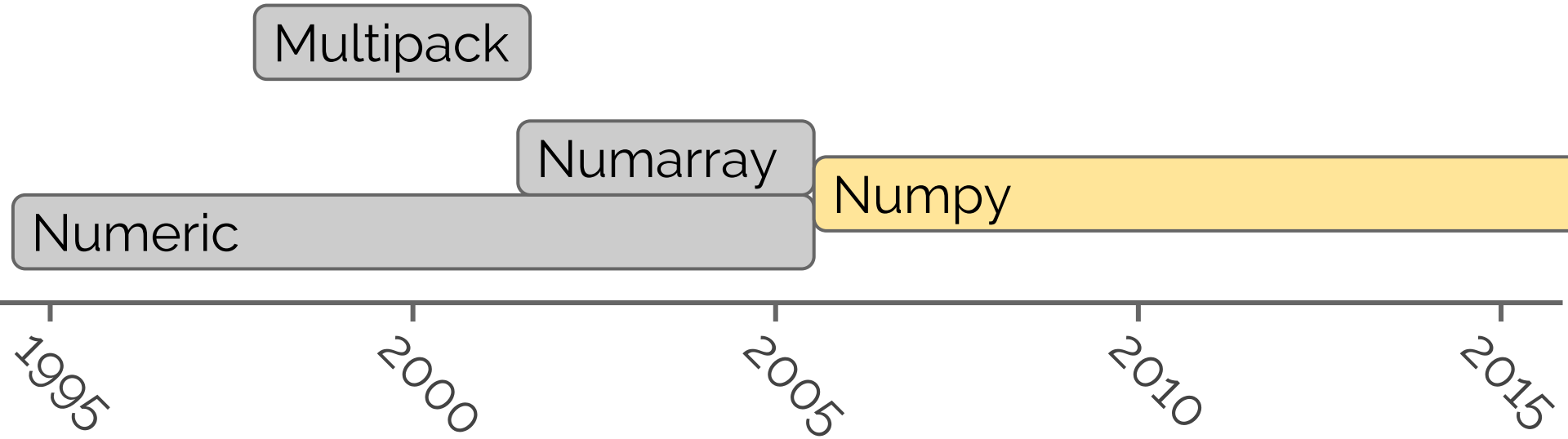
Numeric

1995 2000 2005 2010 2015

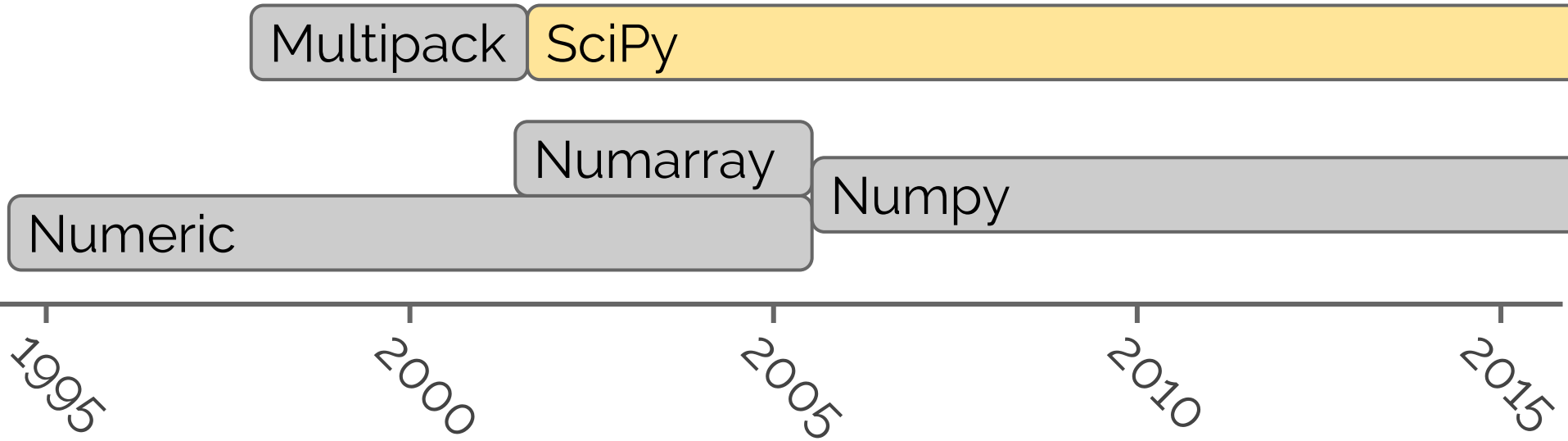
2002: **Numarray**, was created by Perry Greenfield, Paul Dubois, and others to address fundamental deficiencies in Numeric for larger datasets.



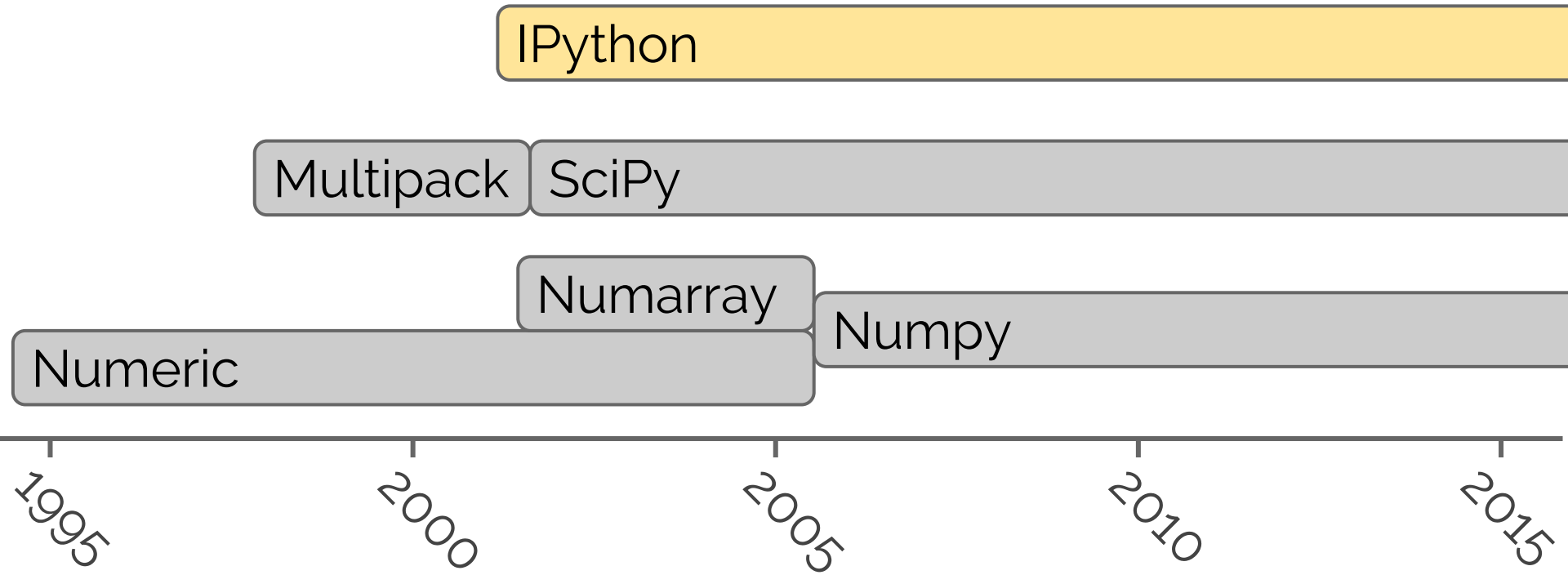
2006: In a Herculean effort to head-off this split in the community, Travis Oliphant incorporated best parts of Numeric + Numarray into **NumPy**



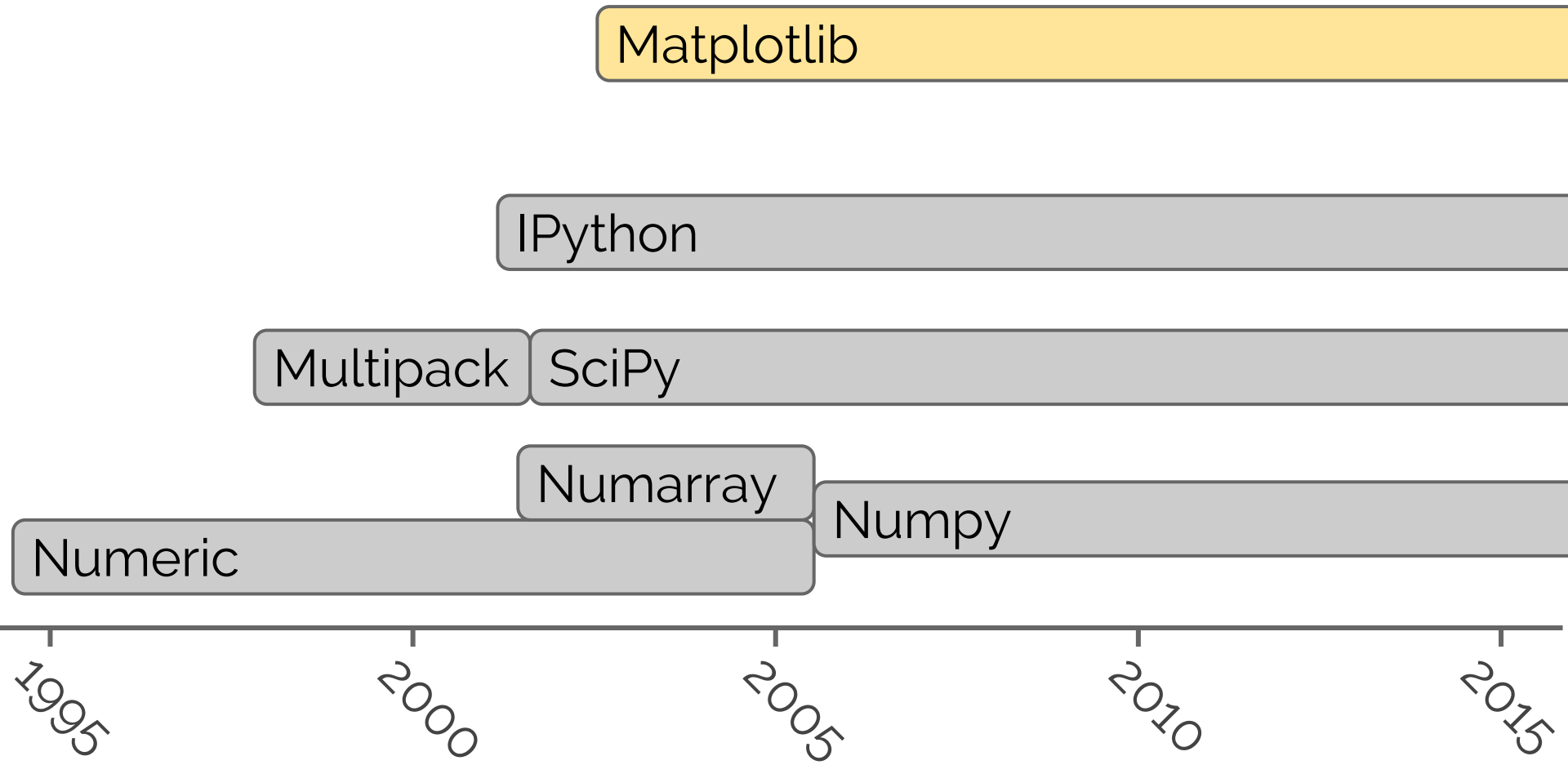
2000: Eric Jones, Travis Oliphant, Pearu Peterson, and others spun multipack into the **SciPy** package, aiming for a full Python MatLab replacement.

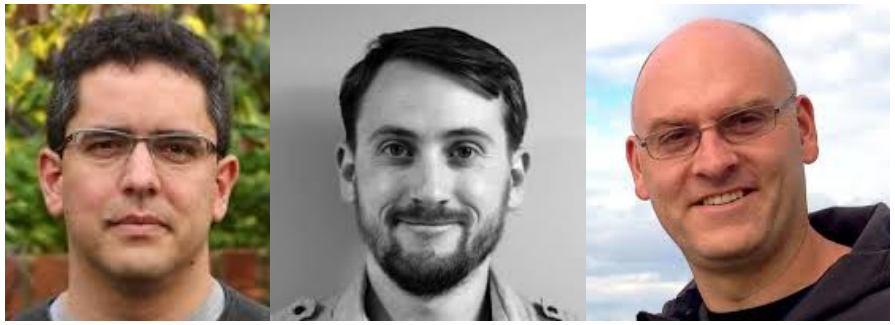


2001: Fernando Perez started the **IPython** project, aiming for a mathematica-style environment for Scientific Python.

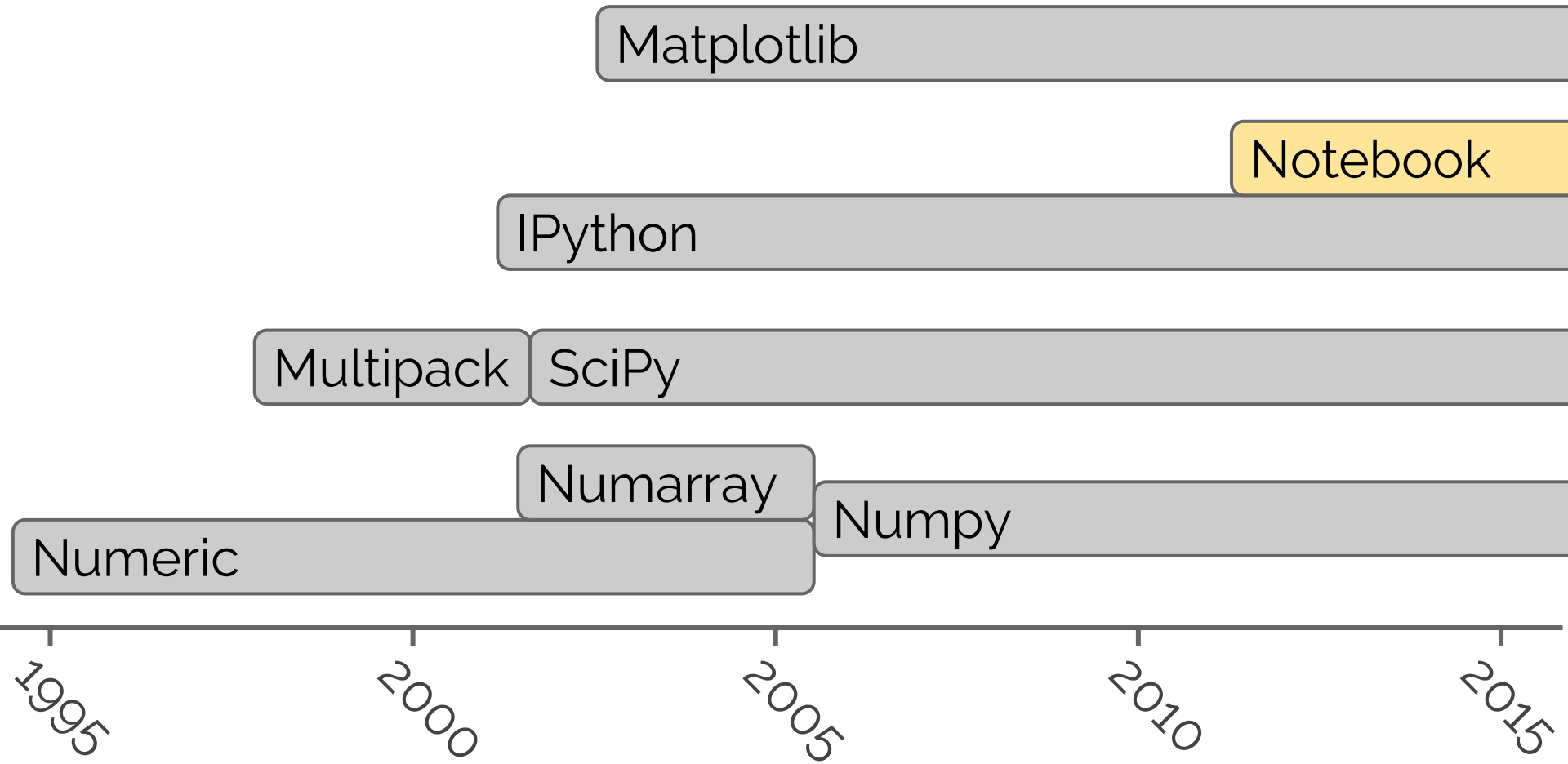


2002: John Hunter wanted an open MatLab replacement, and started **Matplotlib** as an effort at MatLab-style visualization.

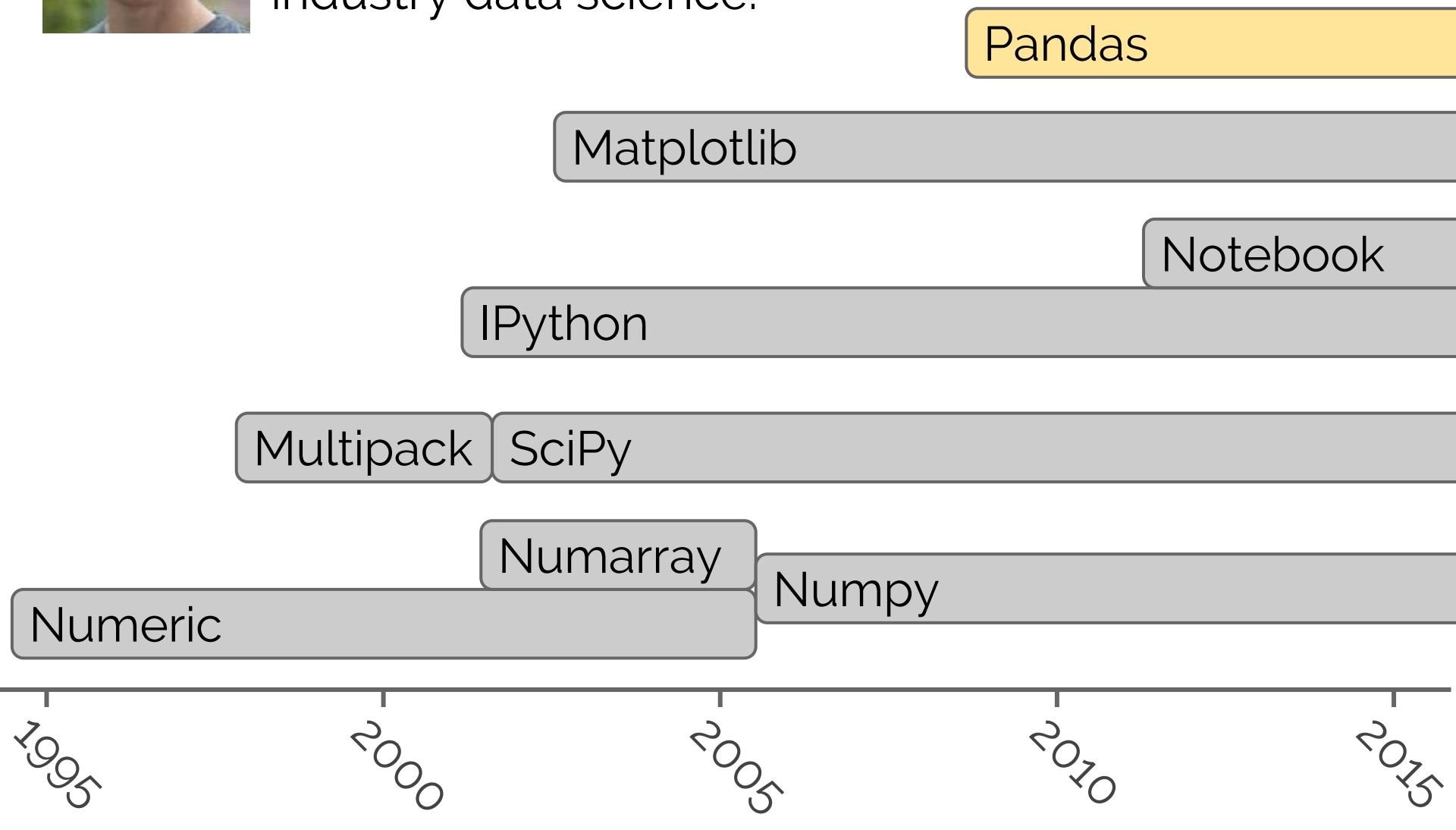




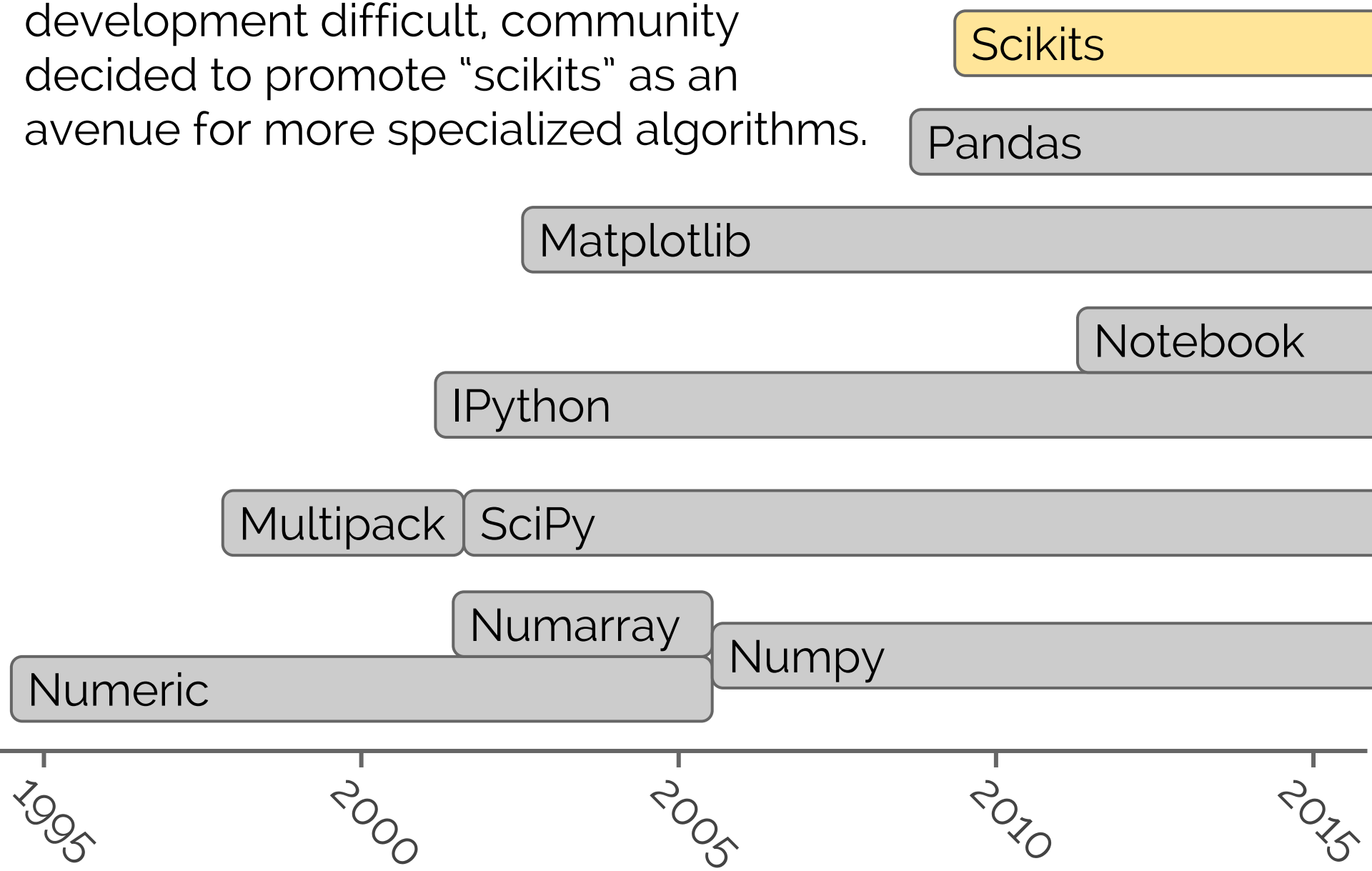
2012: The IPython team released the **IPython Notebook**, and the world has never been the same



2009: Wes McKinney began **Pandas**, eventually drawing-in a *much* larger Python user-base, especially in industry data science.

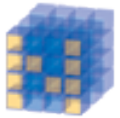


2009: With SciPy's sheer size making fast development difficult, community decided to promote "scikits" as an avenue for more specialized algorithms.



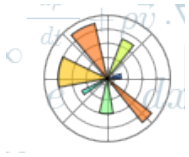
2012: Continuum releases **conda**, a package manager for scientific computing.

# Scientific Python is Federated . . .

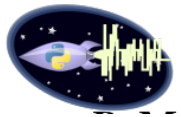



pandas

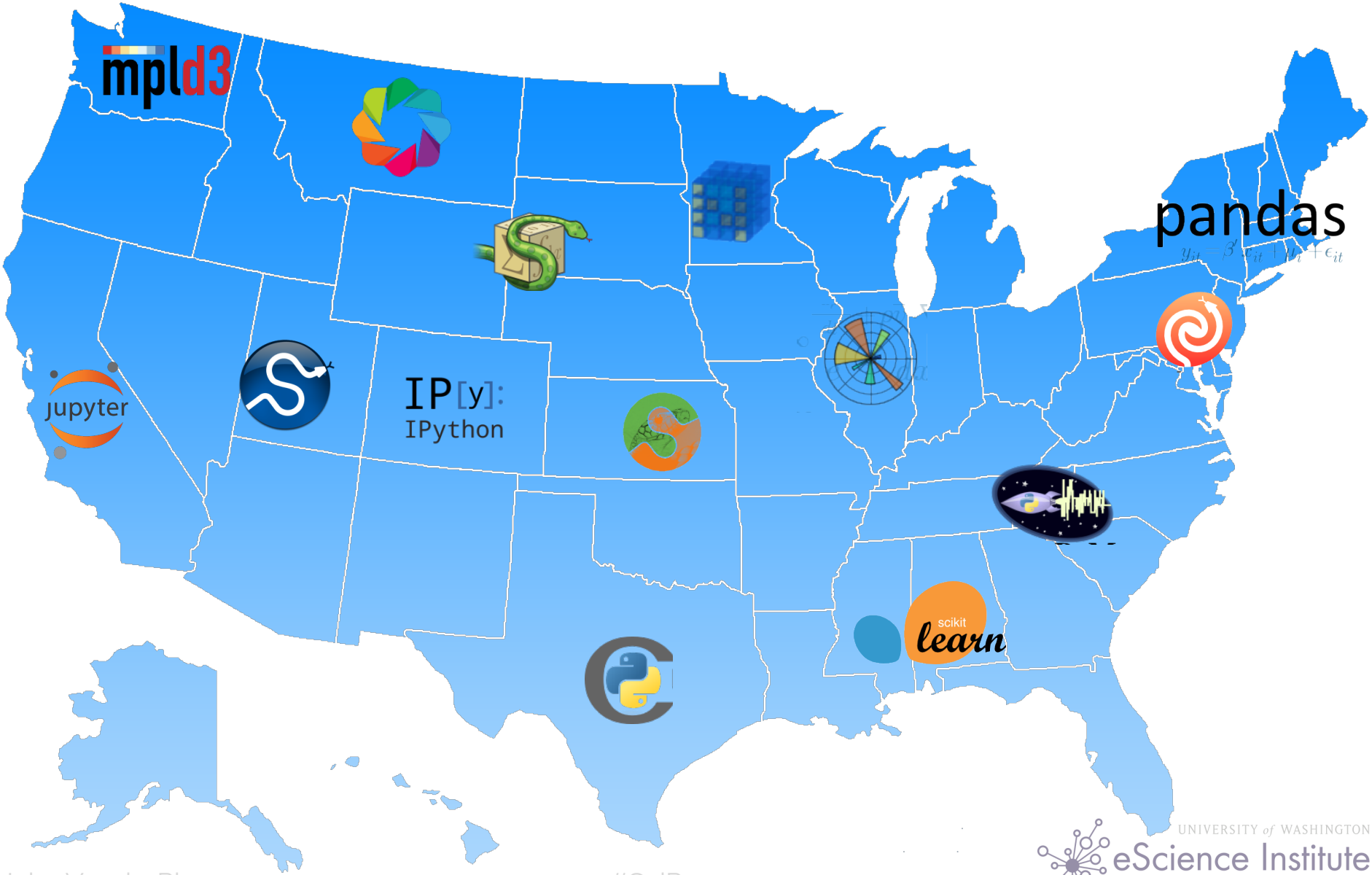
$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



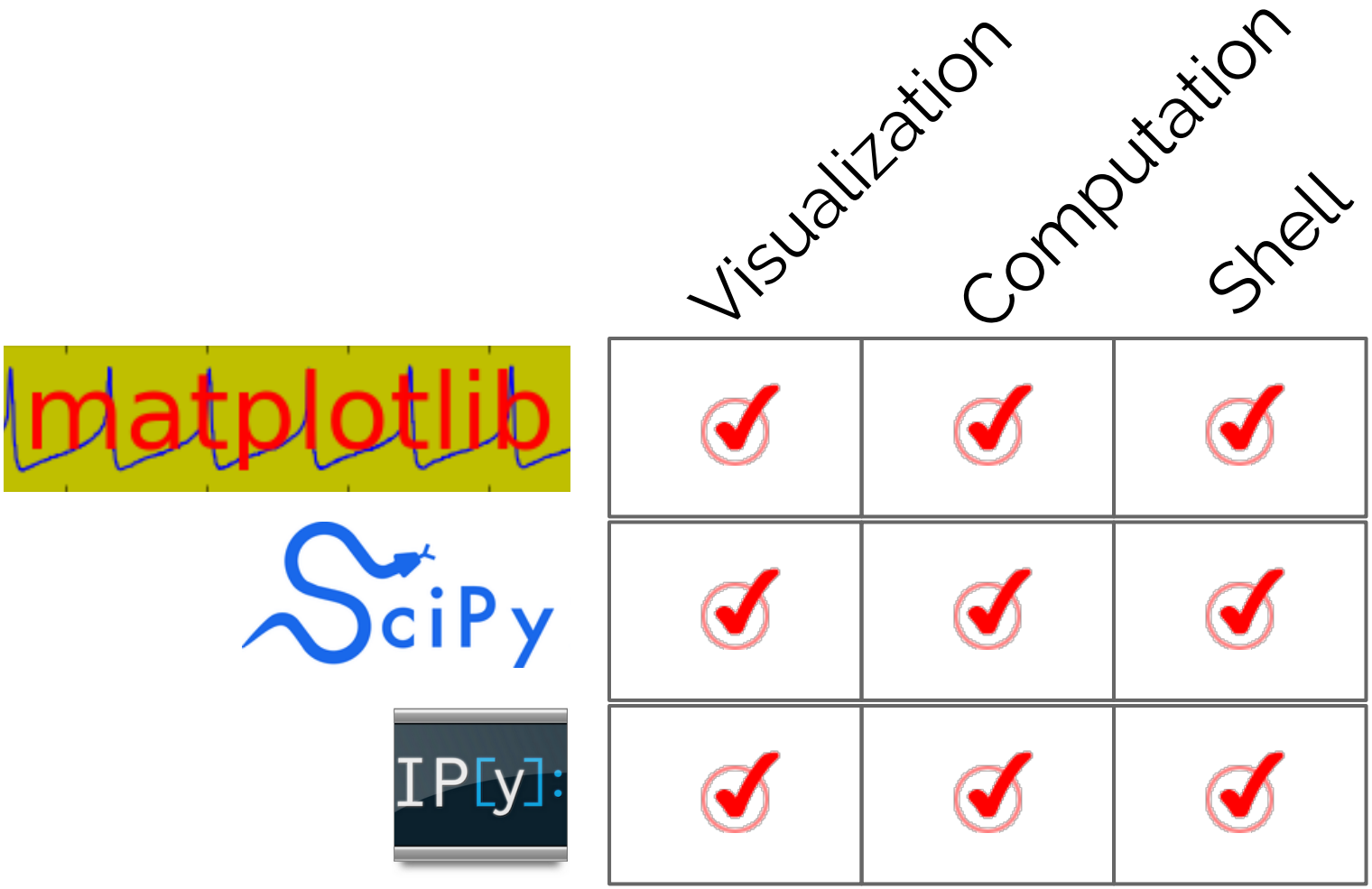

IP[y]:
IPython



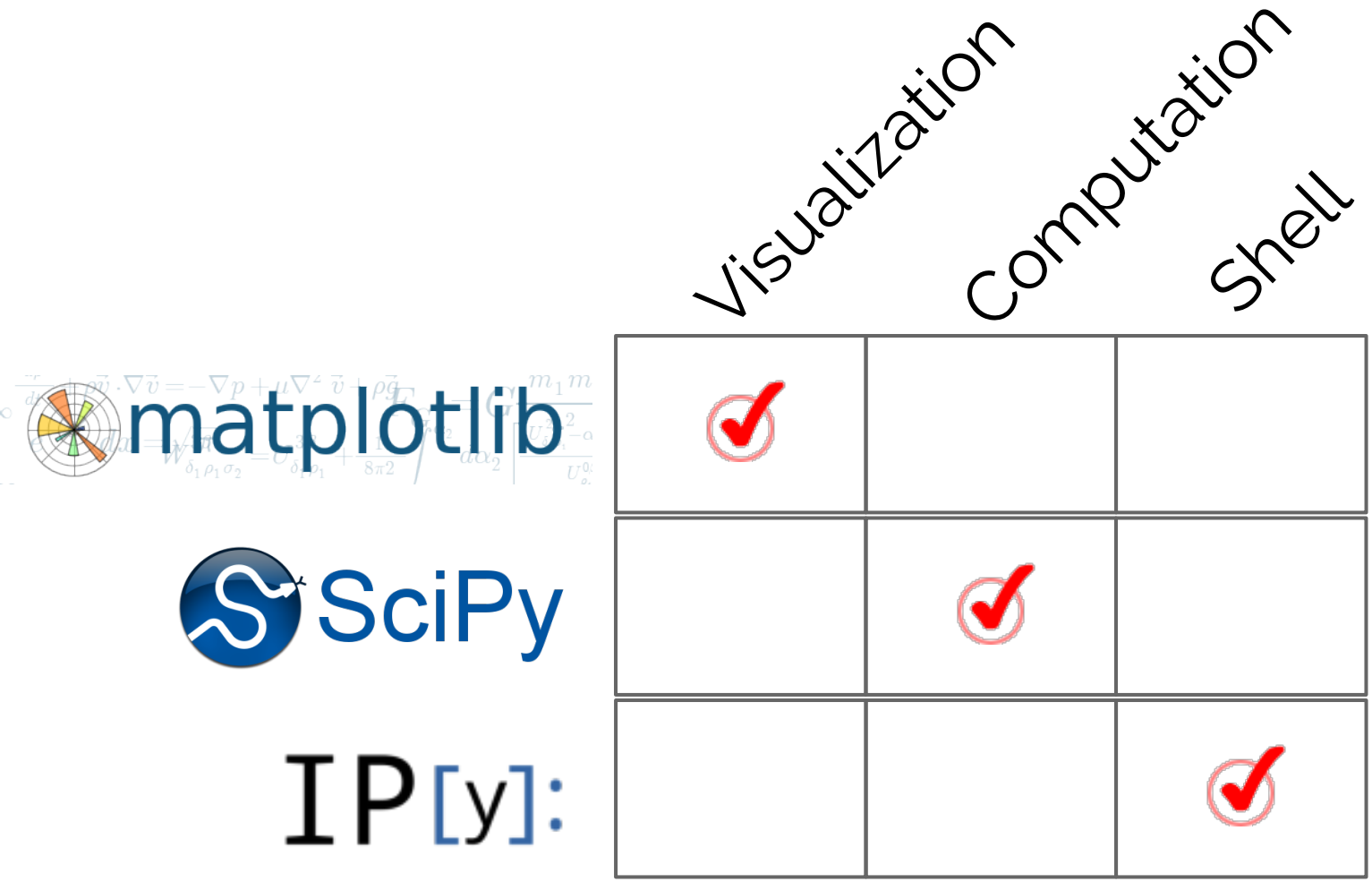
# Scientific Python is Federated . . .



# Early on, developers were simply writing tools to solve their problems . . .



# It took deliberate collaboration to arrive at today's *coherent* ecosystem:

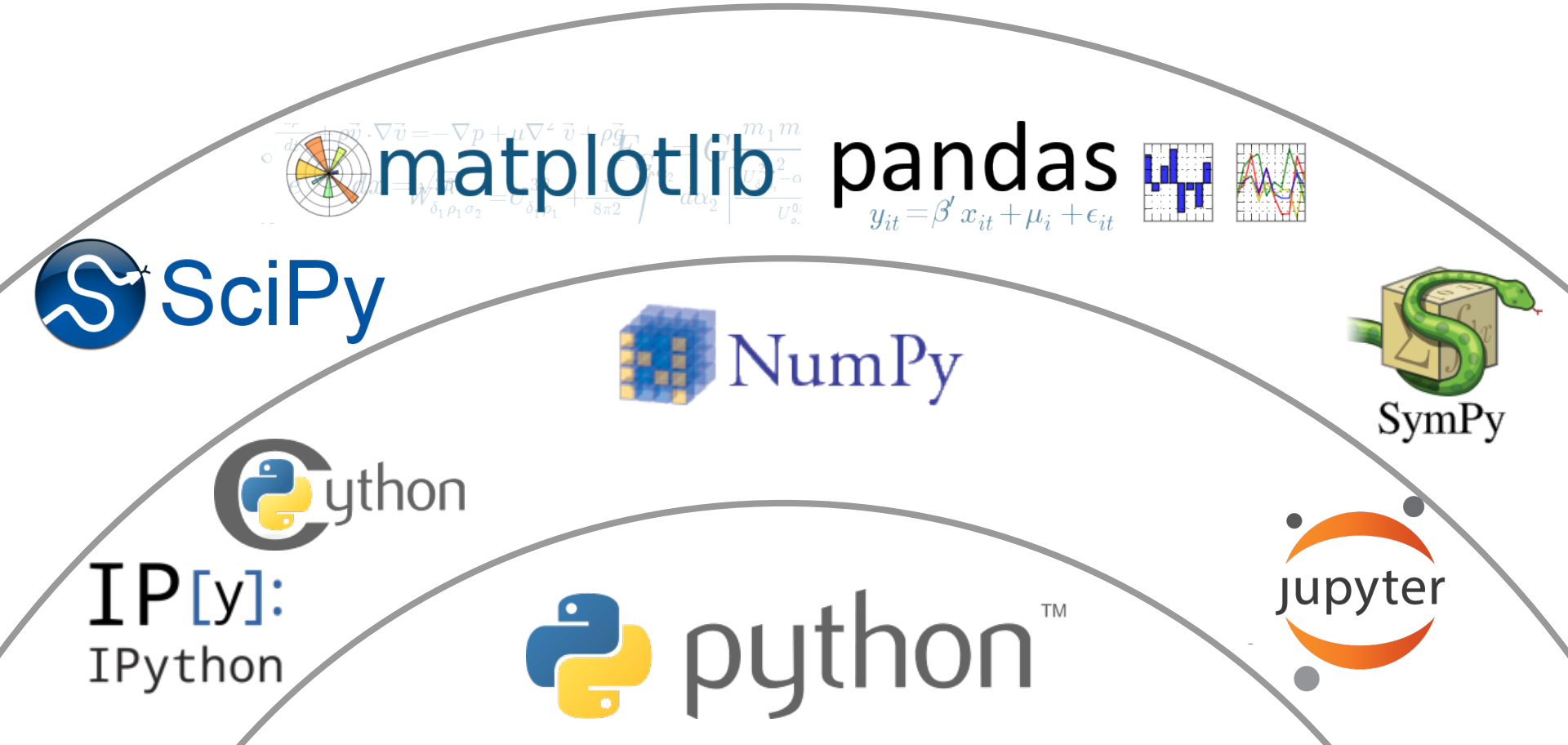


# Toward the Future

# Lessons Learned

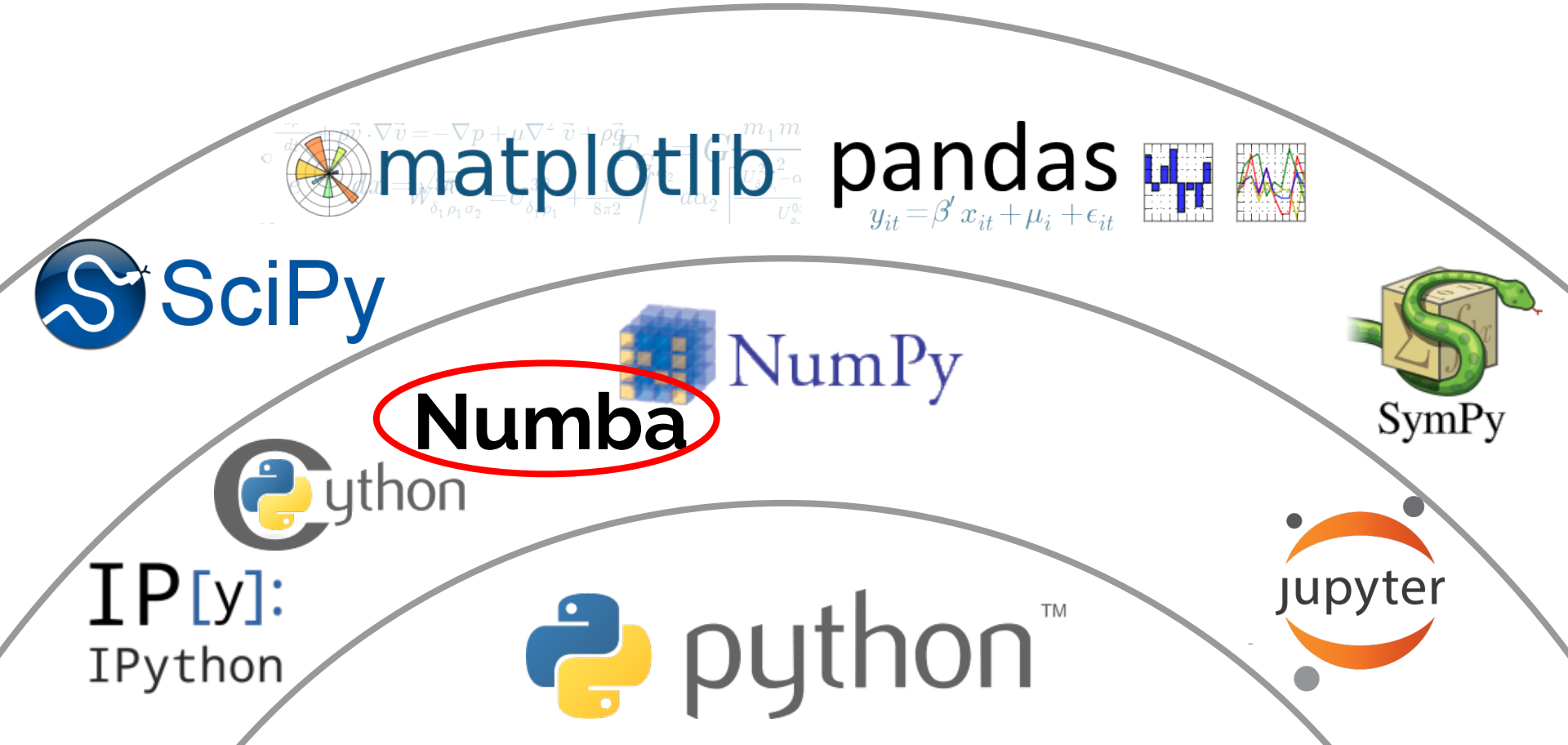
1. No centralized leadership! What is "core" in the ecosystem evolves & is up to the community.

# Evolving Computational Core: Numba?



# Evolving Computational Core: Numba?

Just as Cython matured to become a core piece, perhaps Numba might as well? How might a JIT-enabled scipy, sklearn, pandas, etc. look?



# Lessons Learned

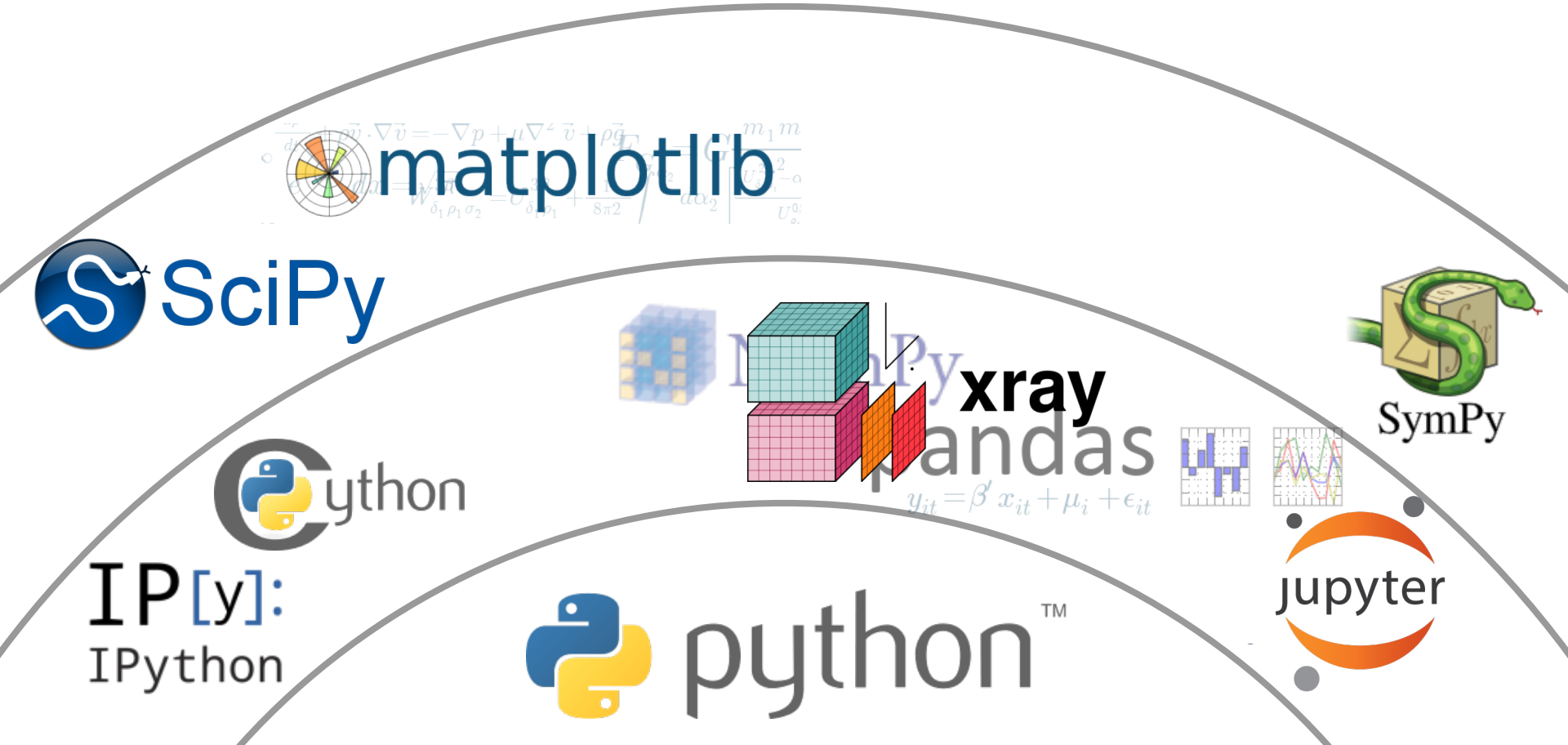
1. No centralized leadership! What is "core" in the ecosystem evolves & is up to the community.
2. To be most useful as an *ecosystem*, we must be willing for packages to adapt to the changing landscape.

# Evolving Computational Core: Pandas?

Modern data is sparse, heterogeneous, and labeled, and NumPy arrays don't measure up: **Let's make Pandas a core dependency!**

# Evolving Computational Core: Pandas?

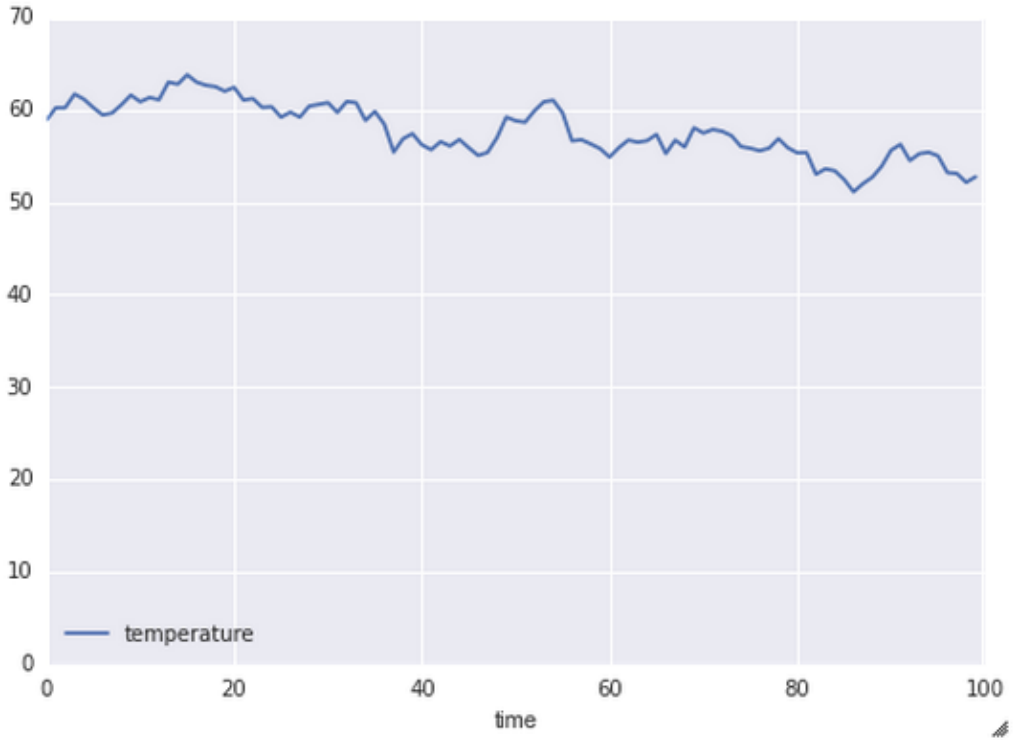
Modern data is sparse, heterogeneous, and labeled, and NumPy arrays don't measure up: **Let's make Pandas a core dependency!**



# Imagining Pandas-enabled Matplotlib:

Old Way:

```
plt.plot(data['time'], data['temperature'])
plt.xlabel('time')
plt.legend(['temperature'])
```



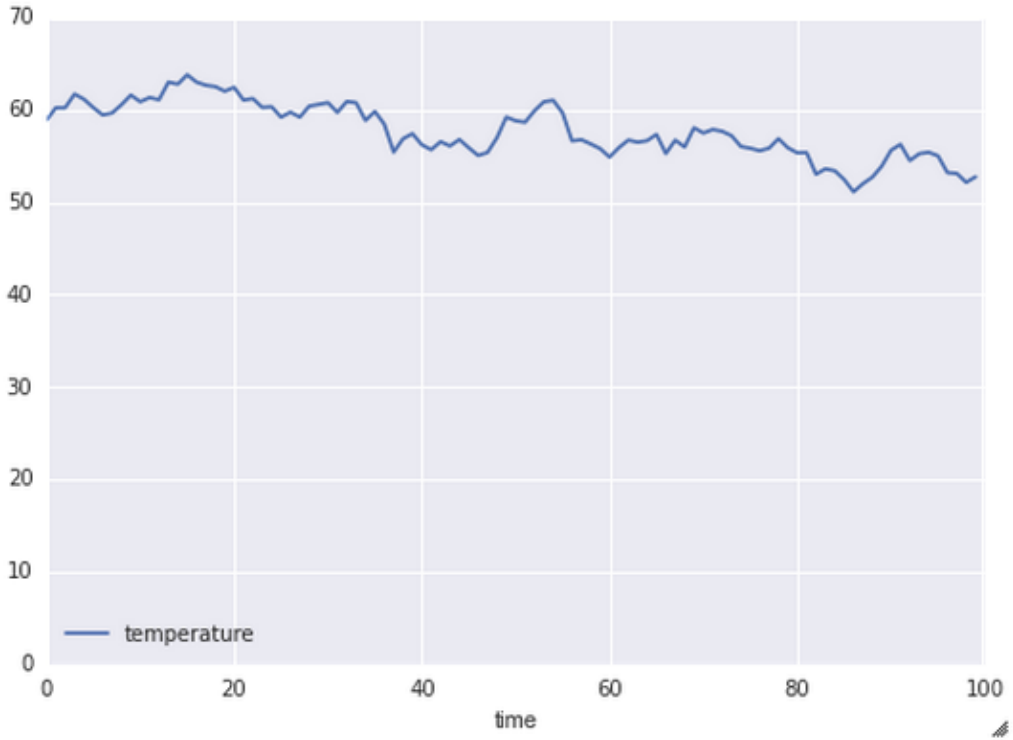
# Imagining Pandas-enabled Matplotlib:

Old Way:

```
plt.plot(data['time'], data['temperature'])
plt.xlabel('time')
plt.legend(['temperature'])
```

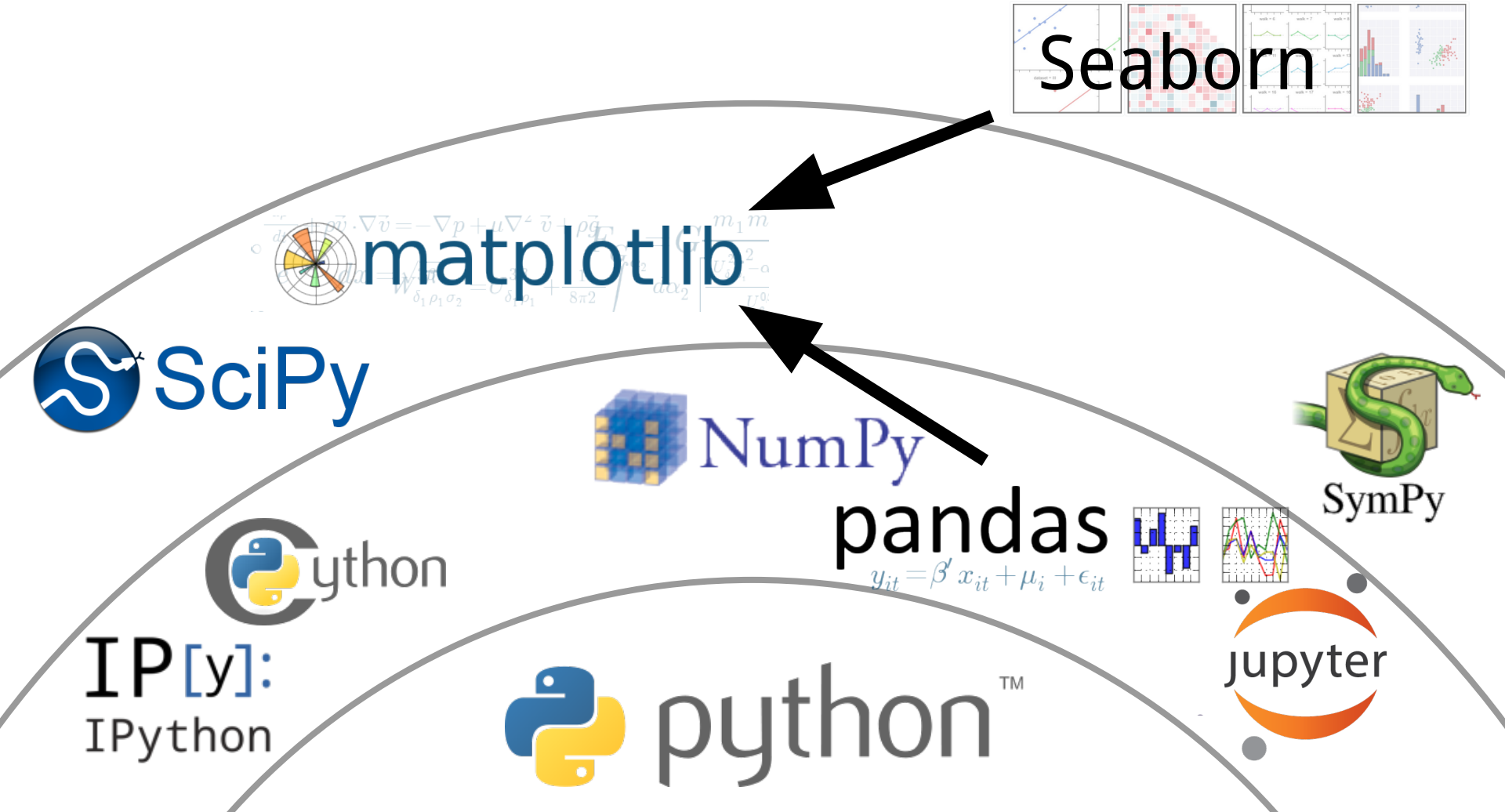
New Way?

```
plt.plot('time', 'temperature', data=data)
```



# Evolving Computational Core

With Pandas core dependency, what elements of Seaborn & Pandas could be moved into matplotlib?



# Evolving the core: SciPy

SciPy's monolithic design was driven by packaging & distribution difficulties.

"In 2001… we came out with what we called the Scipy *library*, but it was really the SciPy *distribution*. I didn't realize it at the time, but that's really what it was: a collection of tools with a single installer, so you get everything up and running quickly.

**- Travis Oliphant**, *EuroSciPy 2014 Keynote*

With `conda`, do we still need a single SciPy package? Should it be split-up into smaller packages?

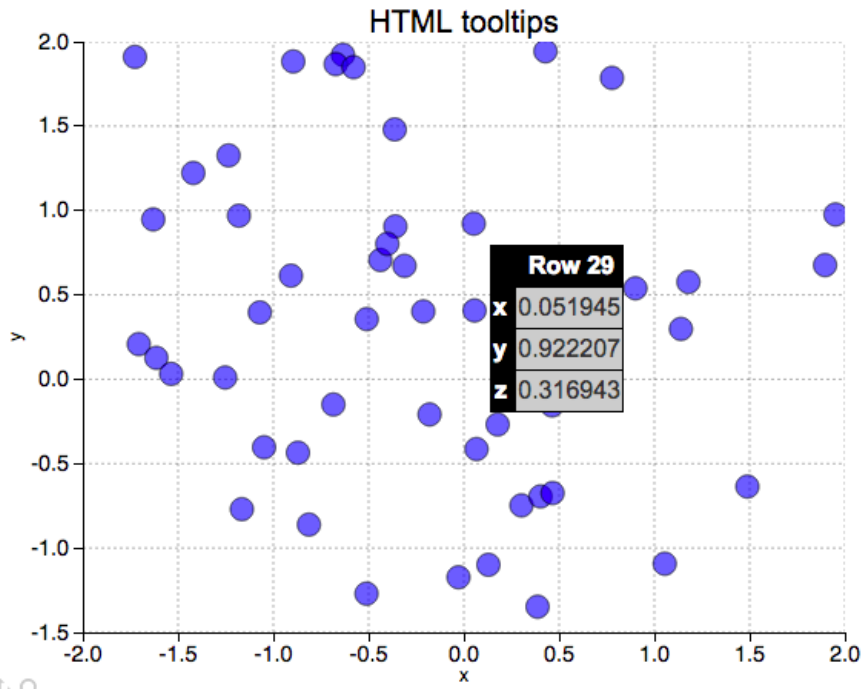
# Lessons Learned

1. No centralized leadership! What is "core" in the ecosystem evolves & is up to the community.
2. To be most useful as an *ecosystem*, we must be willing for packages to adapt to the changing landscape.
3. Interoperability with core pieces of other languages has been key to the success of the SciPy stack (e.g. C/Fortran libraries, new Jupyter framework).

# Universal Plotting Serialization?

Much of modern interactive plotting (d3, HTML5, Bokeh, ggvis, mpld3, etc.) involves **generating & processing plot serializations**.

mpld3




matplotlib

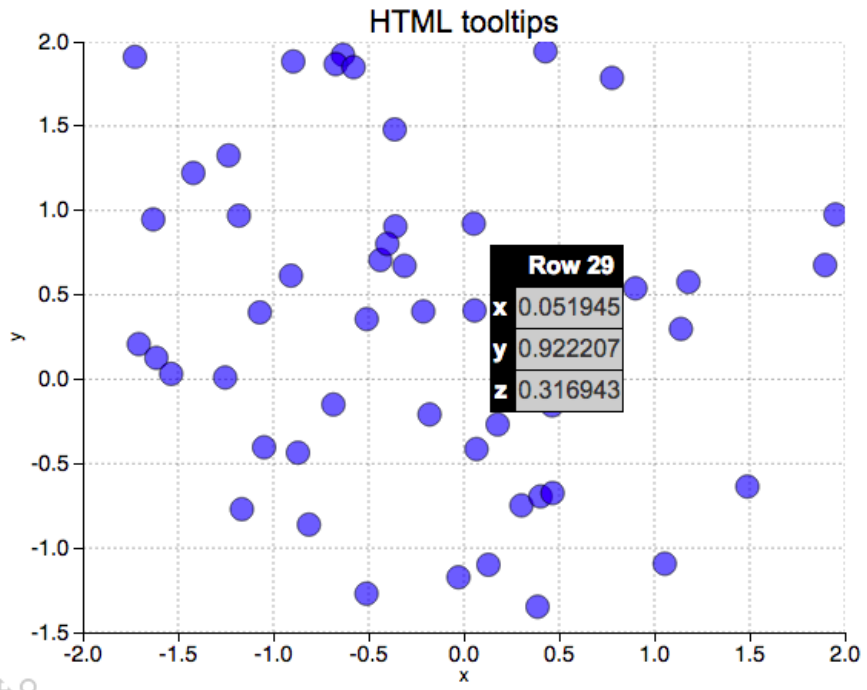
{JSON}

JavaScript

# Universal Plotting Serialization?

Much of modern interactive plotting (d3, HTML5, Bokeh, ggvis, mpld3, etc.) involves **generating & processing plot serializations**.

Doing this natively in matplotlib would open up extensibility!

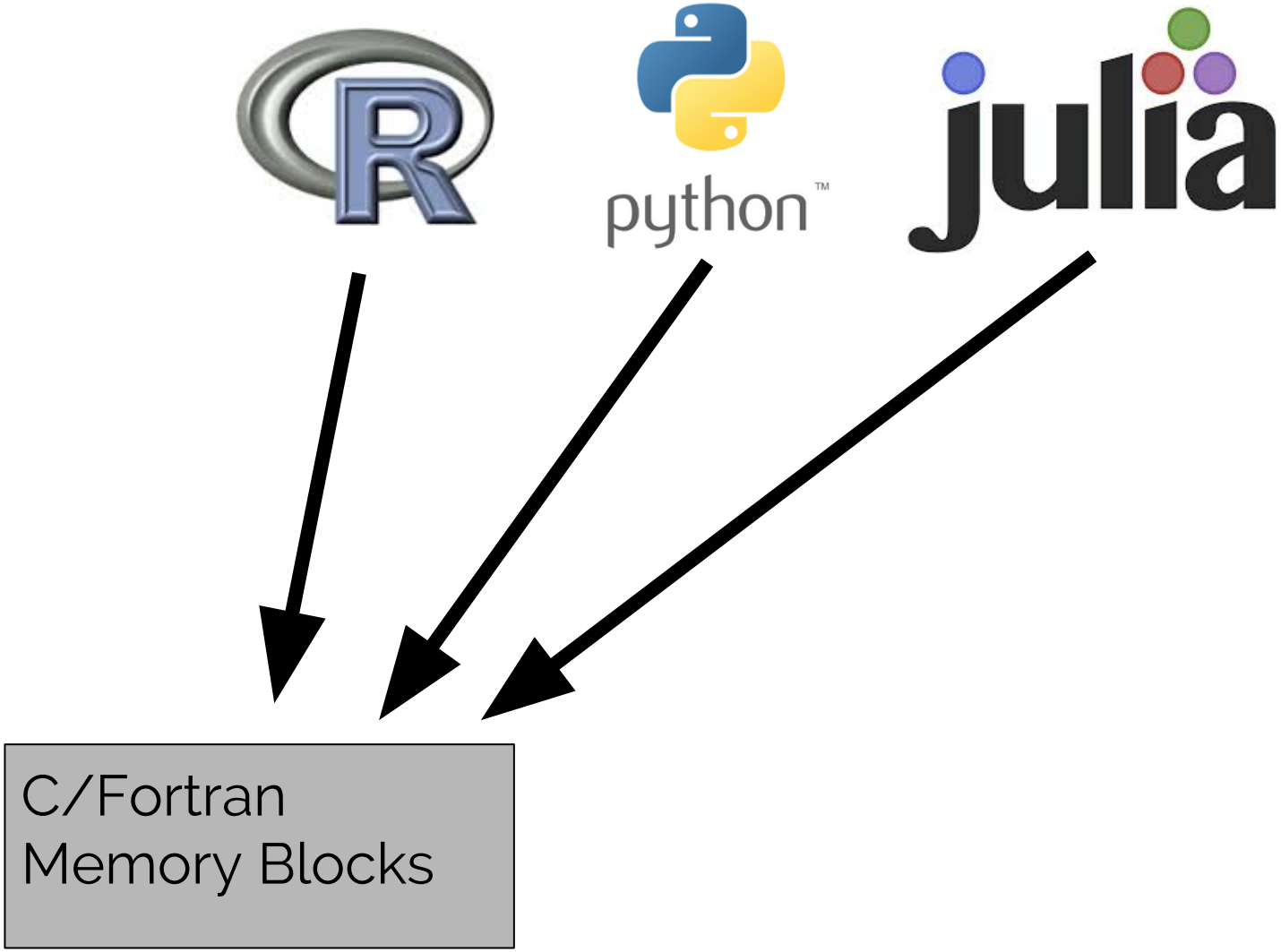



matplotlib

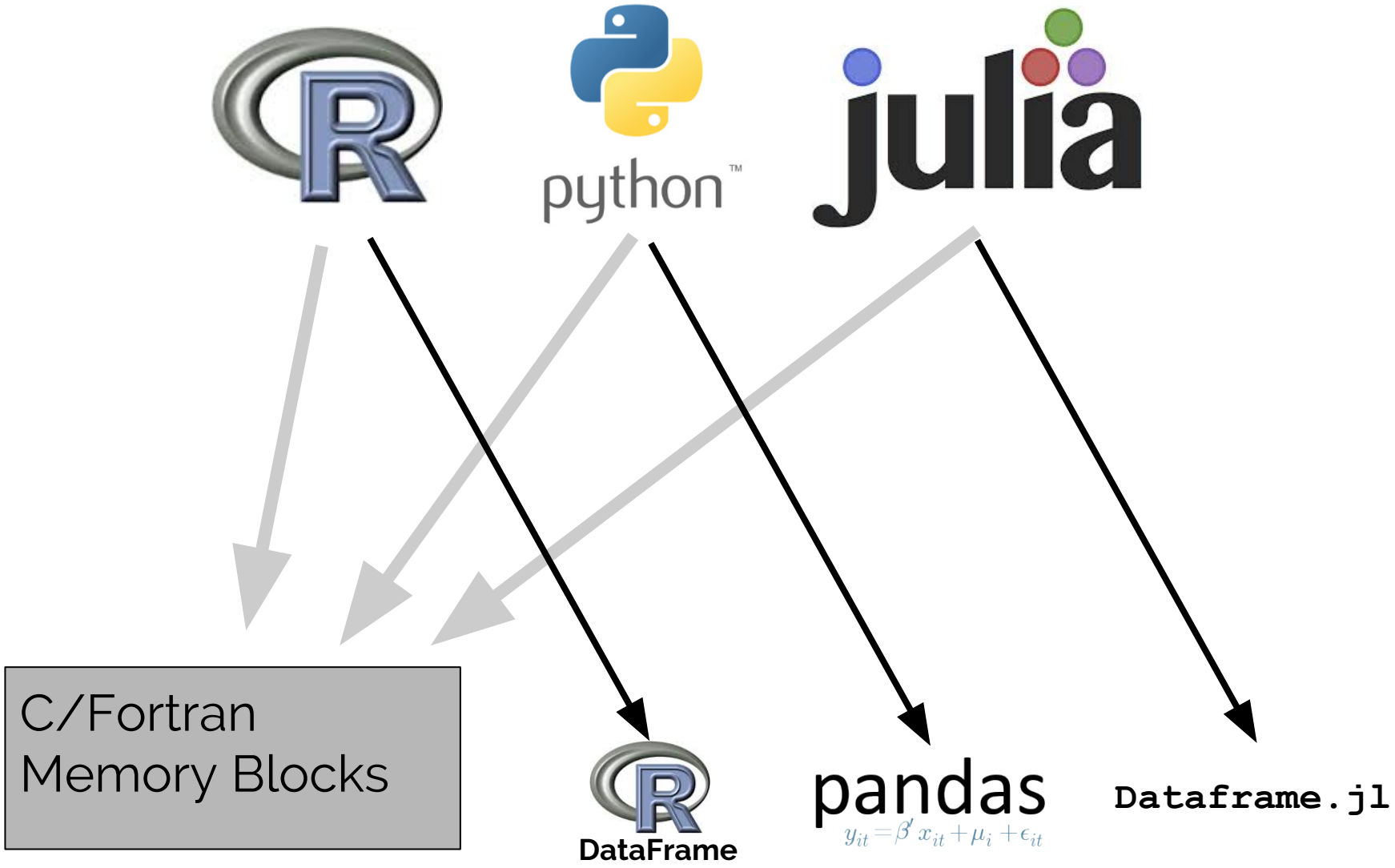
{JSON}

JavaScript

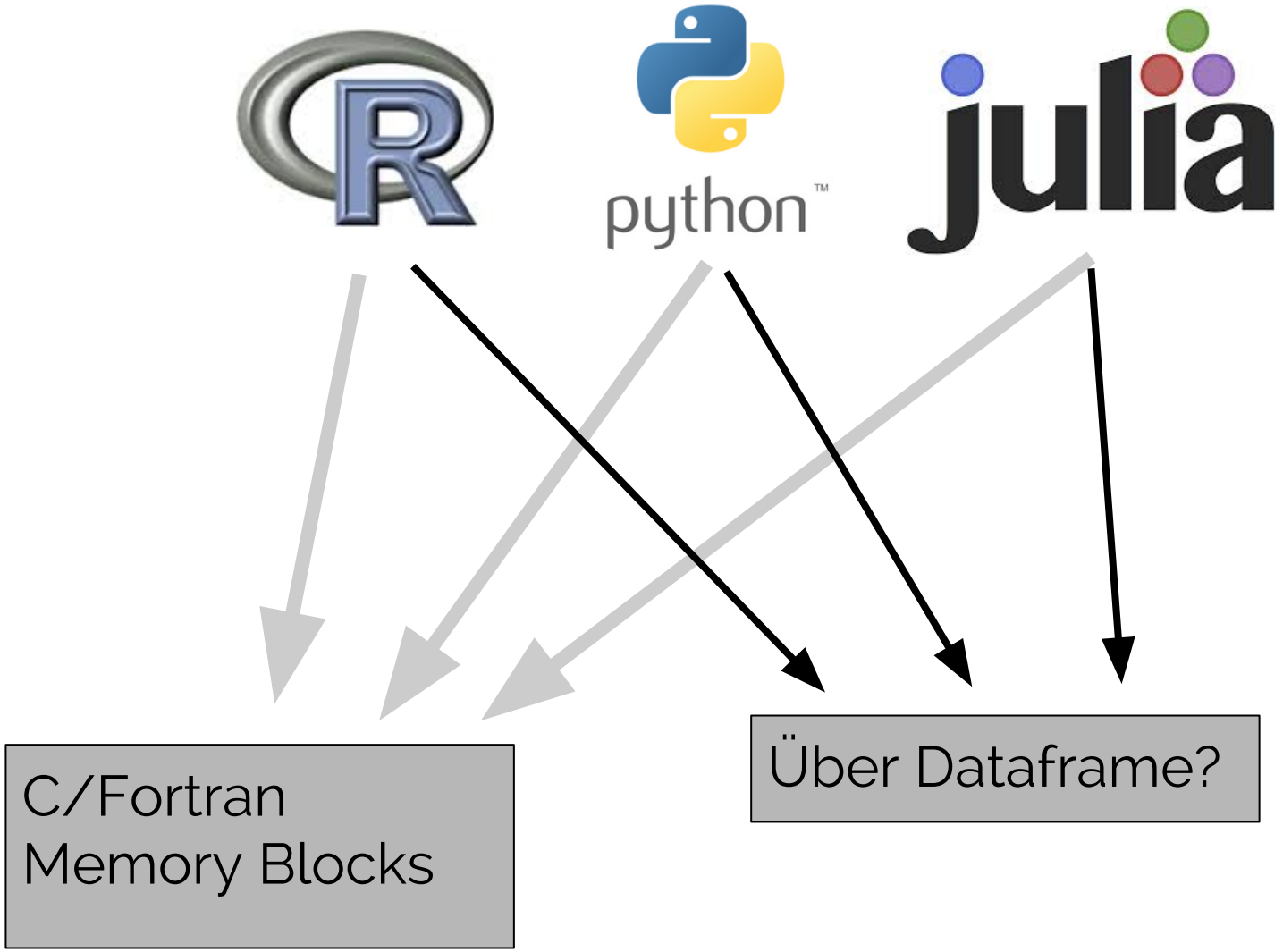
# Universal DataFrames?



# Universal DataFrames?



# Universal DataFrames?



# Lessons Learned

1. No centralized leadership! What is "core" in the ecosystem evolves & is up to the community.
2. To be most useful as an *ecosystem*, we must be willing for packages to adapt to the changing landscape.
3. Interoperability with core pieces of other languages has been key to the success of the SciPy stack. (esp. C/Fortran libraries, new Jupyter framework).
4. The stack was built from both continuity (e.g. Numeric /Numarray→NumPy) and brand-new efforts (e.g. matplotlib, Pandas). Don't discount either approach!

# Considering the Future of Matplolib

Usual complaints about Matplotlib:

- Non-optimal stylistic defaults
- Non-optimal API
- Difficulty exporting interactive plots
- Difficulty with large datasets

# Considering the Future of Matplolib

Usual complaints about Matplotlib:

- ~~Non-optimal stylistic defaults~~ Matplotlib 2.0!
- Non-optimal API
- Difficulty exporting interactive plots
- Difficulty with large datasets

# Considering the Future of Matplolib

Usual complaints about Matplotlib:

- ~~Non-optimal stylistic defaults~~ Matplotlib 2.0!
- ~~Non-optimal API~~ Seaborn, ggplot!
- Difficulty exporting interactive plots
- Difficulty with large datasets

# Considering the Future of Matplolib

Usual complaints about Matplotlib:

- ~~Non-optimal stylistic defaults~~ Matplotlib 2.0!
- ~~Non-optimal API~~ Seaborn, ggplot!
- ~~Difficulty exporting interactive plots~~
- Difficulty with large datasets

Serialization to mpld3/Bokeh/etc?

# Considering the Future of Matplolib

Usual complaints about Matplotlib:

- ~~Non-optimal stylistic defaults~~ Matplotlib 2.0!
- ~~Non-optimal API~~ Seaborn, ggplot!
- ~~Difficulty exporting interactive plots~~
- Difficulty with large datasets

Serialization to mpld3/Bokeh/etc?

**Lesson from Numeric/Numarray, etc.:**

Stick with matplotlib & modify it! (e.g. serialization to VisPy? Numba-driven backend? new backend architecture?, etc.)

**Lesson from Pandas & Matplotlib, etc.:**

Start something from scratch; features will draw users! (e.g. VisPy, Bokeh, Something new?)

# The State of the Stack is up to You.

# ~ Thank You! ~

Email: jakevdp@uw.edu

Twitter: @jakevdp

Github: jakevdp

Web: http://vanderplas.com/

Blog: http://jakevdp.github.io/

"[SciPy] was intended to be a Matlab replacement and so needed libraries, plus, plotting, and a shell to work in, along with tools to integrate with C/C++."
- **Travis Oliphant** *via email*

> "We were trying to plug the major holes in the Python world that made it inferior to Matlab for scientific computing... [Plotting] was probably the biggest obvious gap between Python and Matlab, so we thought that this would be a central piece of SciPy."
> - **Eric Jones** *via email*

> "... from the very beginning, Mathematica and its notebooks (and the Maple worksheets before) were in my mind as the ideal environment for daily scientific work."
> - **Fernando Perez**
> *The IPython Notebook: A Historical Retrospective*

photo courtesey of Fernando Perez