**Open Ended Problems**

**PyLab - https://scipy.github.io/old-wiki/pages/PyLab**

[PyLab](https://scipy.github.io/old-wiki/pages/PyLab226f.html?action=fullsearch&context=180&value=linkto%3A%22PyLab%22)

Currently this page reflects the vision of [KeirMierle](https://scipy.github.io/old-wiki/pages/KeirMierle.html), and not necessarily the community as a whole. By integrating consensus from mailing list discussions, I will refine and polish this vision and form a plan of action such that the community can move the numpy+scipy+ipython+matplotlib ensemble closer to the vision outlined below.

THIS IS NOT COMPLETE YET

See the following post for further discussion of the difference between the vision for a new PyLab expressed on this page, and the existing pylab package which is part of matplotlib:

[http://matplotlib.1069221.n5.nabble.com/pylab-tp23420p23423.html](https://scipy.github.io/old-wiki/external.html?link=http://matplotlib.1069221.n5.nabble.com/pylab-tp23420p23423.html)

**The PyLab Vision**

To make PyLab an easy to use, well packaged, well integrated, and well documented, numeric computation environment so compelling that instead of having people go to Python and discovering that it is suitable for numeric computation, *they will find PyLab first and then fall in love with Python.*

The philosophy behind this vision is to consider Rails and Ruby; while Ruby was somewhat popular beforehand, it was Rails which propelled it to the forefront.

At the moment, the current combination of Python, **[NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html), [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html), Matplotlib, and IPython** provide a compelling environment for numerical analysis and computation. **Unfortunately**, for those who are not already familiar with Python and the intricacies of how to build your own Python environment, or for those not familiar with the details of how there are conflicting names exported by different modules, or how the best list of [NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html) examples is found on the wiki in a non-obvious place (and that the docstrings are not the best documentation), or that the speed of linear algebra operators is dependent on a carefully compiled combination of LAPACK, ATLAS, and Goto BLAS, or a host of other reasons (some outlined below), **the picture is not nearly so rosy.**

**Short-term Goals**

**Documentation** - Dramatically enhance the standard documentation by consistently adopting the new [DocstringStandard](https://scipy.github.io/old-wiki/pages/DocstringStandard.html) for all functions in the [NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html) API.

**API Consistency** - Create an official API for the PyLab system such that there is an official way to import the PyLab packages, and such that there are not multiple functions with very similar names in different packages.

**Installation** - Make the installation process trivial, especially for, e.g. people without root access or spare time.

**Build Process** - Make the build process simple for the combination of the five core components (Python, [NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html), [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html), Matplotlib, and IPython).

**A simple user story**

Joe is frustrated with Matlab, because he finds it is slow when running his neural network experiments. He hears about PyLab from a friend, who recommends it as an alternative.

He finds PyLab as the first search result on Google

He finds a page with **minimal clutter**, showing a couple pictures of PyLab, and a direct download link to a binary for his operating system. He notes that pylab.org must have determined he is running Linux automatically. The page also has a small number of **big, clear links** to promotional materials (screencasts, testimonials), documentation, and community information (how to get involved).

After downloading, the program installs with no hassles, and Joe can launch Pylab by typing 'pylab' and pressing enter in a terminal. Joe is happy that there was no hassle over dependencies on his older university computer, and that installing directly into his home directory (he does not have root access on the university computers) is not a problem.

PyLab notices that it is the first time it is run, and suggests he read the tutorial, and provides a link.

Joe clicks the tutorial link, which his terminal automatically pops up in a browser. The tutorial covers the basics of PyLab, explaining some of the philosophy. The tutorial is clearly written, and covers the basics of array computation and 2D graphing.

When Joe is implementing code, he finds the interactive help invaluable, provided by typing any object or function with a '?' after it in the interactive prompt. The documentation has **copious examples and helpful pointers** to other functions which may be useful (See also:).

Joe implements and runs his neural network simulations, and manages to speed them up by using one of the several methods of optimizing computation in the tutorial. He is so pleased with the results he suggests to his instructor that the entire class should switch to PyLab, as it is free and as far as Joe can tell, **superior in almost every way to MATLAB.**

There are some details omitted here (such as in step 3, does Joe untar the downloaded file or is it an executable?), but those are not the point. The point is that PyLab is a compelling, integrated, usable and superior alternative to MATLAB.

**Why the PyLab name? Isn't that already taken by Matplotlib?**

PyLab should be the name of the entire suite, and I feel strongly that the correct way to import the entire core PyLab API should be via

from pylab import \*

This should include the core parts of numpy, scipy, and matplotlib. This should also be the default namespace set up when the program is launched interactively via 'pylab'. Whether the other components (such as numpy.linalg.\*) should be included in this import is up for debate.

**1. Revamping the Documentation**

For now this section only talks about docstrings, and ignores the other forms of documentation (tutorial, guide, etc). If you want to fill out this section please do!

**Docstring standard**

Travis Oliphant posted a [draft docstring standard on January 10th 2007](https://scipy.github.io/old-wiki/external.html?link=http://article.gmane.org/gmane.comp.python.scientific.devel/5572) which is the basis for the following proposed standard, which is now the [DocstringStandard](https://scipy.github.io/old-wiki/pages/DocstringStandard.html) page. Please look at it.

**Available modules in docstrings**

Examples in docstrings are extremely valuable. However, it is currently never the case that the docstrings in either [NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html) or [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html) use any of the functionality offered by matplotlib. This in unfortunate, especially in the case of [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html), because often the clearest way of demonstrating a function is to plot something.

While it is true that there is an argument which says that [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html) should not become dependent on matplotlib, it appears that this dependence already exists for all intents and purposes. It is likely that only in the most extreme cases one would want to use [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html) without matplotlib. Furthermore, the dependency would only be in the case where a user is executing docstrings in the interactive interpreter; in this case, it is highly likely that the user is doing something which requires some sort of plotting package regardless.

**2. Fixing API consistency**

The current PyLab ensemble has issues with API consistency, mainly stemming from Matplotlib's compatability layer. For example, **consider the load function**, which exists in both [NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html) and Matplotlib:

>>> from pylab import \*

>>> load.\_\_doc\_\_

'\n Load ASCII data from fname into an array and return the array.\n\n ....'

>>> from numpy import \*

>>> load.\_\_doc\_\_

'Wrapper around cPickle.load which accepts either a file-like object or\n a filename.\n '

This function isn't matched! They are clearly different, yet each accepts a filename, and each will break in mysterious ways when one is expecting the functionality of the other. This is especially hard to track down when one is editing a script and running it where the script's import order causes numpy.load to be present, but in the interactive terminal the user has open pylab.load is the exposed function. **This is bad.**

Another example is the confusion around the min and max functions overwriting the builtins, then breaking in weird and unexpected ways. It appears this is now sorted out, with numpy.amax and numpy.amin being the array versions, with numpy.min and numpy.max new names for numpy.amin/amax. I feel as though from numpy import \* should import min and max, but import a min and a max that throw an exception!

Here's a list of conflicts between [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html) and Matplotlib:

[Toggle line numbers](https://scipy.github.io/old-wiki/pages/PyLab)

[1](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_1) In [12]: import pylab

[2](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_2) In [13]: import scipy

[3](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_3) In [14]: p=set(dir(pylab))

[4](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_4) In [15]: s=set(dir(scipy))

[5](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_5) In [16]: k=p.intersection(s)

[6](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_6) In [17]: conflicts = [f for f in k if getattr(scipy, f) is not getattr(pylab, f)]

[7](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_7) In [18]: import matplotlib

[8](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_8) In [19]: matplotlib.\_\_version\_\_

[9](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_9) Out[19]: '0.87.7' <--- this was a pre 0.90 svn IIRC

[10](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_10) In [20]: conflicts[0]

[11](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_11) Out[20]: 'cumsum'

[12](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_12) In [21]: pylab.cumsum?

[13](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_13) Type: function

[14](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_14) Base Class: <type 'function'>

[15](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_15) String Form: <function cumsum at 0xb6731924>

[16](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_16) Namespace: Interactive

[17](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_17) File: /usr/lib/python2.4/site-packages/numpy/oldnumeric/functions.py

[18](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_18) Definition: pylab.cumsum(x, axis=0)

[19](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_19) Docstring:

[20](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_20) <no docstring>

[21](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_21)

[22](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_22) In [22]: scipy.cumsum?

[23](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_23) Type: function

[24](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_24) Base Class: <type 'function'>

[25](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_25) String Form: <function cumsum at 0xb774d5a4>

[26](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_26) Namespace: Interactive

[27](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_27) File: /usr/lib/python2.4/site-packages/numpy/core/fromnumeric.py

[28](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_28) Definition: scipy.cumsum(x, axis=None, dtype=None, out=None)

[29](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_29) Docstring:

[30](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_30) Sum the array over the given axis.

[31](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_31)

[32](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_32) In [23]: conflicts

[33](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_33) Out[23]: ['cumsum', 'ptp', 'fix', 'ravel', '\_\_file\_\_', 'ones', 'rank', 'tri',

[34](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_34) 'insert', 'arange', 'indices', 'loads', 'where', 'mean', 'argmax', 'nonzero',

[35](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_35) 'asarray', 'sum', 'polyfit', 'prod', 'log2', 'power', 'cumproduct', 'corrcoef',

[36](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_36) 'meshgrid', '\_\_name\_\_', 'cov', 'cumprod', 'vander', 'arccos', 'load', 'array',

[37](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_37) 'iterable', 'eye', 'log', 'sometrue', 'alltrue', 'zeros', 'log10', '\_\_doc\_\_',

[38](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_38) 'empty', 'polyval', 'arcsin', 'arctanh', 'linspace', 'typecodes', 'copy',

[39](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_39) 'std', 'fromfunction', 'argmin', 'trapz', 'binary\_repr', 'sqrt', 'take',

[40](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_40) 'product', 'repeat', 'trace', 'compress', 'array2string', 'amax', 'identity',

[41](https://scipy.github.io/old-wiki/pages/PyLab#CA-f2ec97e7012c23e693fcf1ad20eab72c10ce59cb_41) 'amin', 'fromstring', 'average', 'base\_repr', 'reshape']

Most of these are via oldnumeric, but not all. Either way, all oldnumeric functions exposed via pylab shouldn't be.

**The single import statement**

What most users want is for a single import statement to get a consistent set of packages which fulfil most of their needs. This should consist of:

from pylab import \*

That gets them [NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html), [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html), and Matplotlib. A rough equivalent would be:

[Toggle line numbers](https://scipy.github.io/old-wiki/pages/PyLab)

[1](https://scipy.github.io/old-wiki/pages/PyLab#CA-59c0f972206331b5a0529b6a8ceb1a81c13055a5_1) from pylab import \*

[2](https://scipy.github.io/old-wiki/pages/PyLab#CA-59c0f972206331b5a0529b6a8ceb1a81c13055a5_2) from numpy import \*

[3](https://scipy.github.io/old-wiki/pages/PyLab#CA-59c0f972206331b5a0529b6a8ceb1a81c13055a5_3) from scipy import \*

**But there are so many names!**

Not really. from scipy import \* brings in about 20 subpackages (i.e. signal such that you still need to do signal.ifft, but not scipy.signal.ifft) and only 15 new symbols.

**How to fix the API**

Fixing the API will involve a discussion between the core maintainers of each of the core modules (excepting IPython): [NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html), [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html), and Matplotlib.

**3. Fixing installation process**

The installation process has certainly gotten easier over the years; however, the packages in PyLab core should be bundled in a cohesive whole so that the user is not even aware that there are several packages beneath the PyLab label. Certainly, for the busy undergrad who needs to get his signal processing homework done but can't afford MATLAB and doesn't have root access on his university computers it makes sense to have a monolithic binary which bundles everything (ala SAGE).

**4. Fixing the Build process**

Building a basic PyLab setup is straightforward on Debian and Ubuntu thanks to package management, provided one is capable of fishing around for the required build packages. However, a straight forward

python setup.py build

will not produce the most optimized executable; in order to get a highly optimized PyLab system, an elaborate dance is required to compile the BLAS, then Goto BLAS, then LAPACK with that BLAS, then replace a subset of LAPACK with optimized versions from ATLAS. The whole process is entirely unintuitive and, as far as the author can tell, not clearly documented **anywhere**, even for Linux.

**Becoming a better foundation for SAGE**

There is a package called [SAGE](https://scipy.github.io/old-wiki/external.html?link=http://sage.math.washington.edu/sage/) which aims for almost exactly the same goals as PyLab. However, it is even more extreme than the PyLab vision outlined here, because SAGE includes many third party programs for cutting-edge support of symbolic computation. It also makes some incompatible changes to the Python syntax.

SAGE is built from a core of Python, IPython, and [NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html). In a [posting](https://scipy.github.io/old-wiki/external.html?link=http://arcknowledge.com/gmane.comp.mathematics.sage.devel/2006-12/msg00111.html) to the SAGE developer list, the lead SAGE developer, William Stein, described how he wishes [NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html) and [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html) would follow more consistent documentation standards. Shortly thereafter Travis Oliphant committed the documentation standard which should be used in [NumPy](https://scipy.github.io/old-wiki/pages/NumPy.html) and [SciPy](https://scipy.github.io/old-wiki/pages/SciPy.html). **By slowly working the docstring documentation into a consistent state, PyLab can form a more consistent and usable foundation for SAGE.**