Astronomical Data Analysis Software and Systems XI
ASP Conference Series, Vol. 281, 2002
D. A. Bohlender, D. Durand, and T. H. Handley eds.

A Survey of Python Plotting Packages for PyRAF

M. D. De La Peña

National Optical Astronomy Observatory, Tucson, AZ 85719, USA

P. Greenfield

Space Telescope Science Institute, Baltimore, MD 21218, USA

Abstract. In order to provide PyRAF, the Python-based alternative to the IRAF CL, with an environment for interactive analysis and visualization of data, we are evaluating a variety of two-dimensional plotting packages that are currently available for Python. One of the goals is to be able to manipulate data resident in memory in order to provide IDL-like interactive capabilities. We also desire that the package have a well-defined, object-oriented programming interface and design so that it can be extended to accommodate additional capabilities. The real challenge for a plotting package is balancing the requirements of flexibility, which gives the user maximum control, with ease of use. This challenge for the user interface places a great demand on the design and implementation of the system, and requires user feedback to evaluate whether the goal of an easy to use, yet powerful, interactive user interface has been achieved.

This paper lists our requirements for and a description of the plotting packages being investigated for use in PyRAF. We discuss how the best candidates are able to fulfill our plotting tests which are designed to be typical examples of scientific presentation of data. Finally, we discuss our evaluation and experiences and the strategy for the next phase of analysis.

1. Survey

There are a number of plotting packages available for Python in various stages of maturity. No single package at this time provides the full range of capabilities required for the PyRAF system. Fundamentally, we are looking for a package which can serve as a good foundation upon which we can build additional functionality to allow us to achieve our goals. Table 1 presents a list of packages identified for further examination.

2. Evaluation Criteria and Candidates

The evaluation criteria are segregated into non-functional (system-level concerns) and functional capabilities. Some of the items listed as criteria are really

Table 1. List of Python Plotting Packages.

Package	Version	Description
biggles	1.4.0	Object-oriented interface to GNU plotutils
DISLIN	7.5	Commercial library for C, F77, and F90 with a Python interface
pxDislin	0.40	Object-oriented interface for DISLIN
Gnuplot.py	1.5	Object-oriented interface to Gnuplot
graphite	0.2	Object-oriented package dependent on SPING (formerly PIDDLE)
Pgplot.py	03-Nov-1999	High-level interface which uses ppgplot and pgplot
ppgplot	0.11	Interface to pgplot
GDChart	0.6	Interface to a high-performance C library
PyChart	1.10	Library for creating PostScript or PDF charts
plt/gplt	0.1	SciPy plotting libraries: plt is pure Python based on wxPython; gplt is a more full featured library based on Gnuplot

issues for which we need to understand the disposition of the package to ensure that there are no impediments to the implementation of necessary features. The non-functional criteria consist of: redistribution issues and/or license restrictions, object-oriented programming interface, ability to embed the graphics within a widget, "live" graphics when focus returns to the Python interpreter, platform availability, package development and support, existing documentation, installation issues, additional software dependencies, and source code and/or package design availability. The functional criteria are comprised of: interactive mode (screen graphics), support for object graphics, output hardcopy options, well-defined user interface, object-oriented or procedural user interface, ability to manipulate data in memory, stored in files, or computed on-the-fly, ability to render y plots, $x - y$ plots, contour maps, surface plots, and overplotting on images, and control mechanisms for: error bars, axis styles (ticks and labeling), line styles, multiple plotting symbols, font options, color choices, and an interactive cursor.

3. Candidates

Reasons for a package listed in Table 1 not being chosen for detailed examination at this time are: it was not buildable or executable, not maintained, too many additional software dependencies, immaturity of functionality, or not appropriate. Due to the richness of the functionality, maturity of the package, and the ease of installation, biggles, Gnuplot.py, and DISLIN were chosen as the initial candidates to investigate in detail.

3.1. Examples

Figure 1 is a representative plot of wavelength vs flux. While all of the candidates generated a plot of this type, only the plot produced by Gnuplot.py is shown due to space limitations. The biggles and Gnuplot.py are object-oriented systems, and they require a container be instantiated within which a plot will be rendered; the components of the plot are then appended to the container. In contrast, DISLIN is a procedural library so a container is not required. The DISLIN

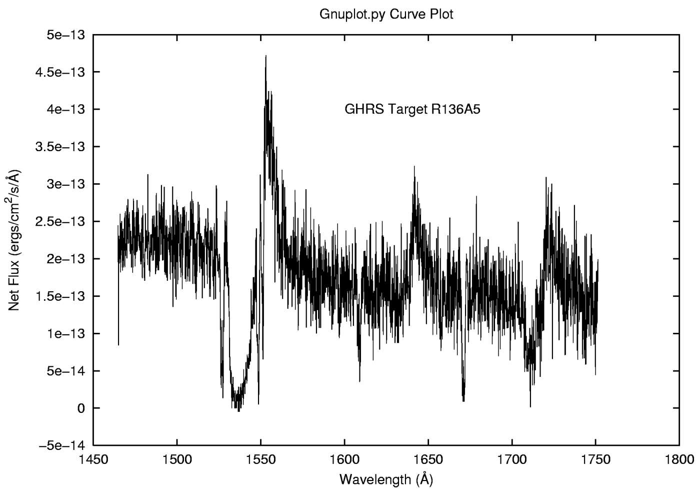

Figure 1. This fully labeled line plot includes annotation, a superscript, and a special symbol, Å. Biggles allows the use of TEX notation for the superscript and angstrom symbol. Gnuplot.py allows the use of the caret symbol for a superscript, but in order to access special characters, the "set encoding iso_8859_1" must be set. DISLIN also recognizes TEX notation for mathematical formulas which includes the superscript; the angstrom symbol in Times-Roman can be generated using the octal equivalent of its ASCII representation.

interface does reflect its heritage in needing not only the arrays to plot, but also the number of points in the arrays.

All of the packages allow the use of various data symbols and line styles, alternative fonts, use of a legend, multiple colors, super and subscripts, and a log axis. However, each package does possess some idiosyncrasies. In biggles, a non-default font is chosen by modifying an account-specific resources file, rather than having a means to change the font locally within the code. While Gnuplot.py can generate plots with color, it is not readily apparent how to control the color settings. For log plots, DISLIN only allows the specification of the log axis to be done in exponents of base 10. DISLIN also requires explicit specification for labels to be generated in exponential format. Support for the generation of contour maps is minimal with biggles and extensive with DISLIN. In Gnuplot.py contour maps are associated with 3-D plotting so to generate a contour map, various settings must be turned off, and the view of the contour map re-oriented; this a computationally intensive and slow process.

4. Evaluation

Two simple timing tests were defined to evaluate the responsiveness of the packages, particularly when dealing with large amounts of graphical data as is often the case in astronomical analysis. The Python package candidates are compared to IRAF, a compiled system, and Interactive Data Language (IDL), a commercial interpreted language. The first test consisted of plotting a spectrum comprised of 30,448 data values. The second test consisted of creating a contour map of a 512 × 512 image, rendering 10 levels. For the spectrum plotting test, IRAF, IDL, biggles, Gnuplot.py, and DISLIN were able to generate the plot on the terminal screen in 0.12, 0.16, 7.5, 2.7, and 0.1 seconds respectively. Similarly, the times required to generate the contour map were 1.01, 1.85, 140, n/a, and 2.7 seconds; Gnuplot.py would not generate a contour map on the terminal screen. While longer plot generation times are not unexpected for interpreted languages, the plotting can be optimized by having the more computationally intensive components done in a compiled language which supports the interpreted environment.

Overall, we found biggles very easy to use and the object-oriented interface and design approachable and appropriate for our needs. While biggles lacks some functionality at this time and the documentation could be more thorough, these issues can be addressed. The main drawback with biggles is the slow rendering of data.[1] Gnuplot.py has an inconsistent interface; some plot controls are accomplished via methods or setting of attributes of the main plot object, while other controls are strings sent directly to Gnuplot for processing. Gnuplot commands are "programmer-oriented" and not considered user-friendly by this evaluation (e.g., access to special characters or the advice to edit PostScript). Gnuplot.py depends upon the Gnuplot Manual which is extremely thorough, but hard to navigate despite being a hypertext document. Finally, DISLIN is a procedural library comprised of many atomic-level routines which provide a wealth of capabilities. The Python interface provided with DISLIN is a set of wrappers to the DISLIN functions. A high-level Python library to DISLIN is the preferred interface. While one does exist, pxDislin, it would not work with the latest versions of Python (v2+). Since the main DISLIN library is proprietary, it would be difficult to extend the capabilities of the package.

5. Summary and Plans

Out of these three candidates, biggles and DISLIN are still under consideration. Since the rendering speed of biggles has been addressed to some extent in a more recent release, biggles will need to be evaluated more rigorously against our other criteria. If the problem with pxDislin can be resolved, this could provide the level of user interface desired, while still having access to the atomic DISLIN functions. Alternatively, graphite released a new version (v0.51), and plt/gplt (SciPy) was released in late August. Due to the importance of the plotting for PyRAF, it will be worth spending some time evaluating these packages. The final choice of a plotting package for PyRAF will not be based just upon which package currently possesses the most functionality. Rather, it is crucial that the package not have any impediments to the implementation of additional capabilities required for PyRAF. For example, none of the packages has much functionality to support images, particularly the display of images with a graphics overlay. Another requirement is the ability to embed the graphics within a widget, and a full screen cursor is necessary for interactive work so the cursor must be able to communicate with the underlying plotting software.

[1] Just after the ADASS XI conference, a test version of biggles was released which is purported to have a dramatic increase in plotting speed.