Astronomy and Computing 13 (2015) 50–57

Contents lists available at ScienceDirect

Astronomy and Computing

journal homepage: www.elsevier.com/locate/ascom

Full length article

Toyz: A framework for scientific analysis of large datasets and astronomical images☆

F. Moolekamp*, E. Mamajek

Department of Physics & Astronomy, University of Rochester, Rochester, NY, 14627-0171, USA

ARTICLE INFO

Article history:
Received 30 June 2015
Accepted 1 October 2015

Keywords:
Big data
Visualization
Python
HTML5
Web application

ABSTRACT

As the size of images and data products derived from astronomical data continues to increase, new tools are needed to visualize and interact with that data in a meaningful way. Motivated by our own astronomical images taken with the Dark Energy Camera (DECam) we present *Toyz*, an open source Python package for viewing and analyzing images and data stored on a remote server or cluster. Users connect to the *Toyz* web application via a web browser, making it a convenient tool for students to visualize and interact with astronomical data without having to install any software on their local machines. In addition it provides researchers with an easy-to-use tool that allows them to browse the files on a server and quickly view very large images (>2 Gb) taken with DECam and other cameras with a large FOV and create their own visualization tools that can be added on as extensions to the default Toyz framework.

1. Introduction

In the past, large scientific datasets were used mainly by large collaborations while independent researchers worked with much more manageable volumes of data. Over the past few years we have been entering a new paradigm where very large sets of data are available to (and at times even generated by) much smaller groups. This abundance of data has highlighted a shortage of scientific tools to store, organize, analyze, and visualize that data. Fortunately this problem overlaps with the needs of the industrial community at large and in the past decade there has been a lot of work by traditional scientists, data scientists, and software engineers to develop software to aid researchers in dealing with this new (and rewarding) problem.

Unfortunately much of the current work in astronomy is often on the fringe of what is possible and has been done before, meaning the types of data we work with poses new challenges, which in turn create a need for new tools (Merenyi, 2014; Gopu et al., 2014; Lins et al., 2013; Loebman et al., 2014; Federl et al., 2012, 2011). Ideally these new tools should be built on existing frameworks that are under active development by software engineers to minimize the effort from research scientists while taking advantage of the latest technologies and updates to existing codes. The Python language has become a fertile ground for rapid software development and with the creation of a vast array of modules for scientific image and data processing like *numpy* (van der Walt et al., 2011), *scipy* (Jones et al., 2001), *pandas* (McKinney, 2010) and *scikit-image* (van der Walt et al., 2014); machine learning modules like *scikit-learn* (Pedregosa et al., 2011); statistics and modeling packages like *scikits-statsmodels*, *pymc* and *emcee* (Foreman-Mackey et al., 2013), and what has become the de facto astronomy python project *astropy* (Astropy Collaboration et al., 2013) and its affiliated packages.

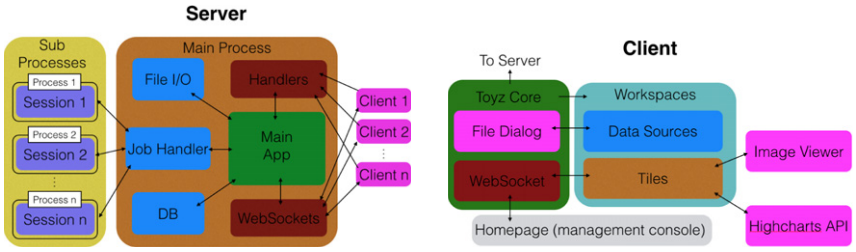
While many of the tools listed above are useful for astronomers, data scientists, and software engineers; there is a great divergence when it comes to tools for visualization. Much of the interactivity and visualization work done in the realm of data science and software development tends to be focused on web frameworks like *jQuery Ui*, *Highcharts*, *D3.js* and even more advanced libraries using webGL like *PhiloGL*, *pathGL* and many others; or R libraries like *ggplot2* (Wickham, 2009). Contrast this with astronomy where programs like *ds9* (Joye and Mandel, 2003) that are used primarily by astronomers with few updates and changes over the past decade. Several recent python packages have been created to help bridge the gap between professional visualization tools and those available in astronomy: *GLUE* (Beaumont et al., 2014) provides a rich GUI for interacting with datasets and images and *Ginga* is one of the most advanced frameworks for viewing and interacting with FITS images.

The disadvantage of using any of the visualization tools in astronomy mentioned above is that to run efficiently all of them

☆ This code is registered at the ASCL with the code entry ascl:1507.006.

* Corresponding author.
E-mail address: fmooleka@pas.rochester.edu (F. Moolekamp).

http://dx.doi.org/10.1016/j.ascom.2015.10.001
2213-1337/



(a) Server architecture. (b) Client architecture.

Fig. 1. Basic architecture of the python web application (server) and HTML5 client. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

must be run on a local machine with data stored locally. With new instruments like the Dark Energy Camera (DECam) that create 2 Gb images (0.5 Gb compressed) and over 1Tb of data products per night (Valdes and Gruendl, 2014; Flaugher et al., 2012), it is no longer feasible to store an entire observing run (or even a single night) on a laptop or PC. Recognizing the need for a server side image viewer several groups have been independently developing web applications to serve images from a remote server to a client with only a web browser installed including *VisiOmatic* (Bertin et al., 2015), *Data Labs* (Fitzpatrick et al., 2014), and now *Toyz*. *VisiOmatic* is an open source web application running on an Apache web server with an IIPImage (Pillay, 2014) server to display large images in a browser using a so called "slippy map" implementation (similar to Google maps). In addition to viewing images, the *VisiOmatic* client also enables users to interact with the image including marking point sources and plotting slices of the data. One of the few drawbacks to *VisiOmatic* is its dependency on libraries and applications not usually run by astronomers including a web server (that must be configured). It also provides no native support for a file dialog (as web browsers do not include a tool to browse a remote file directory), causing users to write one themselves or create a webpage on the server for *each* image they want to view.

When we first began to analyze our own DECam images, which took up over 1 Tb of disk space on our server, we realized that in order to view the images and analyze the catalogs we created with them that we would need a new tool, preferably one that could run on the server storing the data and allow a platform independent way for users to connect to the data and interact with it. This was our initial motivation for creating a new python package called *Toyz*, which seeks to combine the best of all of the software discussed so far: the remote image viewing of *VisiOmatic*, the interactivity of *GLUE* and *Ginga*, the astronomical tools of *astropy*, and the convenience of doing it all in a single framework built on existing Python and HTML5 software maintained by computer scientists. Because *Toyz* is a framework, not an application, it is designed to be easily customized by end users for their specific scientific needs but easy enough to use that a class of undergraduates could use it for analyzing their data without having to install any software on their home computers. One of the guiding principles of *Toyz* is that an undergraduate student should be able to install *Toyz* and begin analyzing data on his/her first day!

In this paper we highlight the various functions of *Toyz*. Section 2 describes the core *Toyz* package that allows users to view images and interactive plots in their browsers, Section 3 describes the affiliated package *Astro-Toyz* that incorporates astronomy specific tools including WCS and interactive tools for the image viewer, and Section 4 describes future plans for integrating *Toyz* with other software packages.

2. Toyz

Toyz can be thought of as a platform-independent tool for visualizing and interacting with large images or catalogs of data. Instead of trying to create a one-size-fits-all application, *Toyz* is designed to be an open source framework that scientists can customize to fit their own research needs.

A graphical representation of the server is shown in Fig. 1(a). The web application at the heart of *Toyz* is built on the Tornado web framework (Darnell, 2015), a python library originally written by FriendFeed as the backend for their social media website. User authentication is done via HTML handlers built into Tornado while most other communications between the server and client are done via WebSockets (Hickson, 2011): a bi-directional protocol that uses an HTML handshake to setup an open communication between the server and the client without the need for constant polling by the client to get the status of a job. Similar technology is used for a variety of websites and web applications including Jupyter (formerly iPython) notebooks (Ragan-Kelley et al., 2014). A separate module handling file I/O provides an API to load data from a variety of formats (see Section 2.2). The file I/O module is written to allow users to create affiliated packages or extensions that allow users to create custom classes for loading additional data types not currently supported by *Toyz* with minimal coding.

Each time a new connection is made to the server a new process called a *session* is spawned using python's multiprocessing module. All of the variables and methods defined in a session will be stored until the user closes the browser and disconnects from the server. This environment is completely separate from the web application (which handles connections to and from the server) where the state of a user's variables are stored for the duration of the connection. Because each session is a separate process, *Toyz* is able to take advantage of all of the processing cores on a server, meaning that if the number of processors scales with the number of simultaneous users, large classes and groups should be able to access the same *Toyz* server with little reduction in performance. To communicate with an associated client each session has a single websocket that connects to a single browser window on remote client, which can send jobs to be run on the server in the form of a JSON object specifying the python module, function and function parameters. All of the jobs sent from a client to the server are verified for authenticity and put in a queue to be run for the correct session. Once a job is completed, a response in the form of a JSON object is sent to the browser that at a minimum contains a status key (indicating whether or not the job completed successfully or encountered an error) and often additional keyword arguments generated by the function.

On the client side all communications are pushed through a single function that maintains information about the current session (a graphical representation of the client is shown in Fig. 1(b)). When initialized the user can choose how errors and warnings that might occur while running a job are handled as well as what actions to take when various types of responses are returned. A file dialog is also initialized that allows users to browse the directory tree on the server, functionality that is not incorporated into web browsers for obvious security reasons. The default homepage when a user logs onto the server is a management console that allows one to

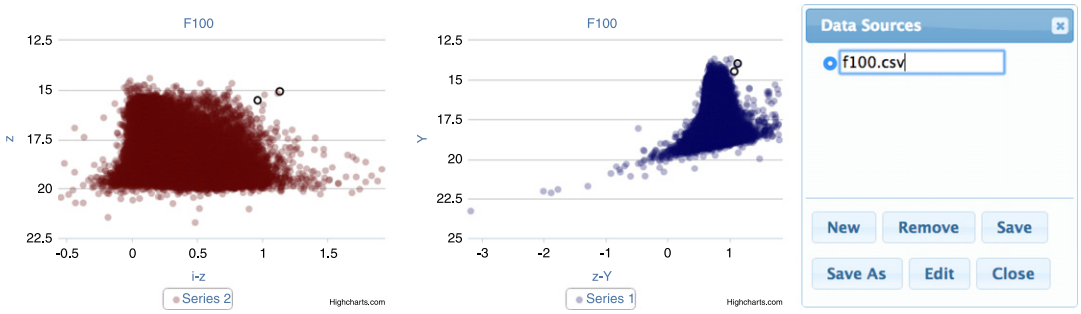


Fig. 2. Workspace with two different color–magnitude plots loaded from the same data source. When points are selected in one of the plots *Toyz* will automatically select the same points in every additional plot generated from the same data source on the server. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

set shortcut paths and allows administrators to set user and group permissions (see Section 2.4).

To assist users in developing their own custom tools a GUI module parses JSON objects (or python dictionaries) to build interactive tools like drop boxes, sliders, and buttons without the need for javascript or css code. All of the menus and controls in *Toyz* have been generated using the same GUI framework, which is thoroughly documented on the website at http://fred3m.github.io/toyz along with several examples.

The remainder of this section discusses additional built-in features of *Toyz*.

2.1. Workspace environment

The main working environment in *Toyz* is referred to as a *workspace*. Graphically a workspace is a blank webpage that allows users to add a collection of customizable tiles. Each tile is associated with some functionality, such as displaying an image or plot, and can be moved and resized in the browser window. The workplace itself is just a JSON object that stores the state of the current users session including the size and location of each tile, links to data sources that the server session and client tiles are connected to, and any other metadata specified by the creator of the tile.

While *Toyz* comes with two default tile types: *Highcharts* plots (see Section 2.2) and image viewers (see Section 2.3), a template is included with the source code to allow users to create their own custom tiles with access to all of the variables of the workspace on the server and the client. In addition, users are able to save the workspace by generating a url that will load a saved workspace for the user and any collaborators with whom the url and permissions are shared. Multiple users can load the same workspace at the same time however they currently cannot share the same session, meaning that if one of them modifies the workspace and saves their changes to the server, all of the other users would have to refresh the page in their browsers to see the changes. See Section 4 for planned upgrades for workspace sharing.

2.2. Data connectivity and interactive plots

The gold standard open source package for data visualization in python is *GLUE*, a python package which allows users to load a series of datasets into memory and provides a GUI for plotting connected datasets and images in tiles on the viewing window. The only current drawback to *GLUE* is that the data sources must be stored locally, which is not always practical for the reasons mentioned earlier. While a future version of *Toyz* seeks to integrate with *GLUE* and extend all of its functionality to the browser, the current incarnation ports some of the most important features, including the ability to create linked plots.

Users are provided an interface to load a variety of data sources onto the server. By default *Toyz* will load any format that is integrated with *pandas* including SQL databases, HDF5, and text files as well as *numpy* binary files and text files that can be opened using standard python I/O functions. Until the user tries to do something with the data (like plot it in the browser) it remains solely on the server, saving time and bandwidth. It is also possible to extend the available file types by adding a custom module, for example the *Astro-Toyz* package extends the available file types to FITS tables, VOTables, and all of the other file formats that can be read from *astropy* Tables.

To interact with the data *Toyz* provides a GUI to *Highcharts*, an open source javascript library that allows users to display interactive plots in a web browser and is free for academic and personal use (and reasonably priced for commercial use). *Highcharts* includes functionality to select data points as well as drill down, zoom into subsets of a plot, and display information about a highlighted point.

Toyz includes an interface to choose columns from a data source loaded on the server and create a plot using a subset of the *Highcharts* API. The user can choose the title, axis labels, tick marks, grids, line styles, marker colors and shapes, and various other features to make plots easier to view without any programming necessary. Each plot is created in a new tile in the workspace and all of the plots connected to the same data source are linked together so that selecting a point (or collection of points) in one of the plots will select the same point(s) in all of the other plots, making it easier to view high dimensional or "wide" data (see Fig. 2). Consistent with the *Highcharts* API, multiple datasets can also be plotted on the same chart. While *Toyz* lacks some of the more advanced features of *GLUE* at the moment, like merging data sets and linking table columns to image axes, it provides a previously unavailable method to quickly explore data stored remotely.

The data source API is modular so that while *Highcharts* is currently the only supported plotting library it would be straightforward to add an interface for other packages like D3.js or webGL support. Any changes made to the data by the Highcharts API or a custom API can be saved to the server using the data source GUI. This tool allows users to save the data as any file type supported by pandas or any affiliated packages (for example FITS binary tables with astro-toyz and astropy).

The processes of loading a single dataset from a server to a client can be divided into three steps:

1. Loading a dataset from a supported file type into memory.
2. Transmitting the data from the server to the client.
3. Plotting the data in the browser using Highcharts.

Of course these steps depend on the speed of the server, the network connection, and the clients processor speed respectively. We performed a set of benchmarks using a server with 8×3.0 GHz Intel Xeon processors running Ubuntu 14.04 and a 1.8 GHz Intel

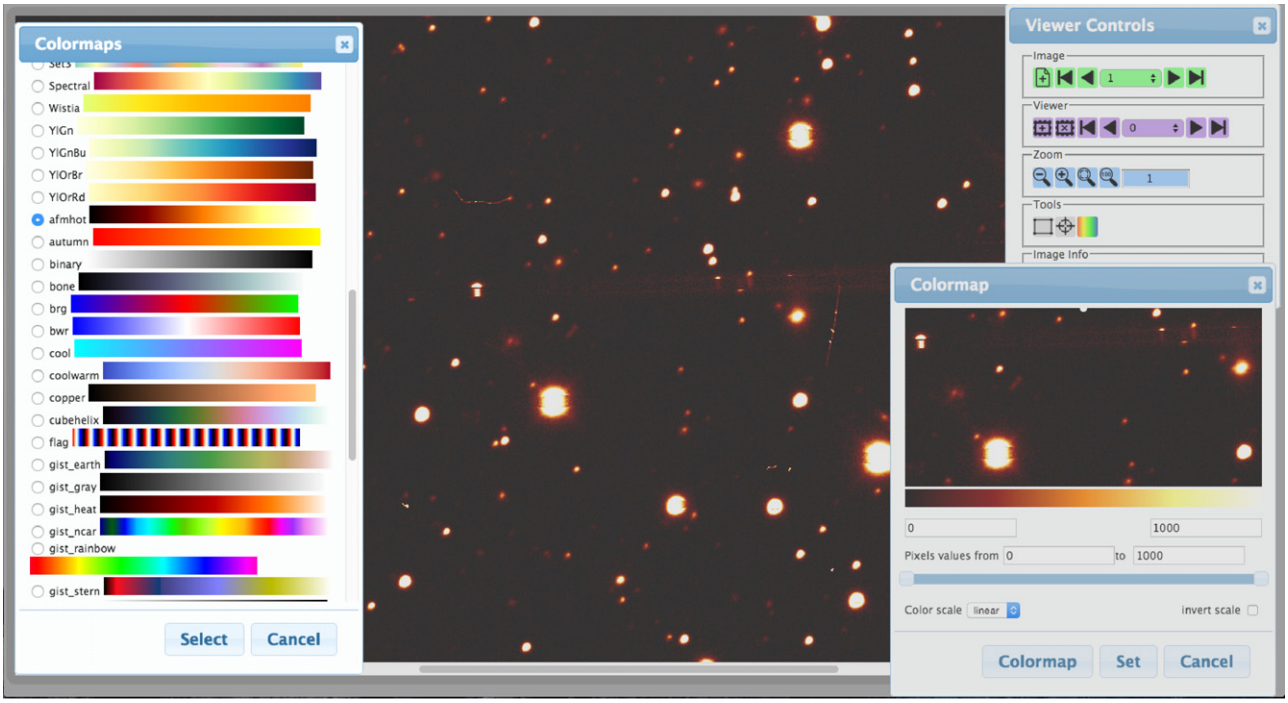


Fig. 3. *Toyz* Image Viewer. For FITS files the full catalog of matplotlib colormaps is available as well as linear and log scaling and inverse color maps. To save upload time only a small tile in the center of the image is used to modify a colormap before it is applied to an entire image. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Core i7 client running OS X to approximate the time needed to load datasets of varying sizes from 10^2 to 10^6 records (see Fig. 4(a) and (b)). Python was able to quickly load a dataset with 10^7 points but transmitting the data to the client caused all 4 browsers we tested to crash (see Section 4 for future upgrades that should make it possible to load over a million points in a browser). While the time to load the data into memory is very small (<1 s), the time to transmit the information over the network and process it in the browser represents the bulk of the time delay. In the case of a 1 Gb/s ethernet connection to a local network (Fig. 4(a)) the majority of the load time is spent plotting the data in the browser while using a much slower 10 Mb/s connection (Fig. 4(b)) causes the total load time to be dominated by data transfer over the network. We also tested the major browsers supported by Toyz: Opera, Chrome, Firefox and Safari (Fig. 4(c)), finding that for small datasets (10^5 points or less) all of the browsers performed about the same while for larger datasets the users browser choice affected the results by a factor of 2.

2.3. Image viewer

The image viewer was initially developed to view collections of large astronomical images that were too large to fit on a local hard drive. Until recently the domain of viewing astronomical images rested on software that had to be installed on a users machine and could only efficiently view images stored locally. Even larger detectors such as MOSAIC (Pogge et al., 1998) at 8K × 8K px and the One Degree Imager (ODI) (Jacoby et al., 2002) at 12K × 12K px produce images small enough that a single nights observations can easily be stored on a notebook or PC and viewed on one of the existing viewers. Newer cameras like the Dark Energy Camera (DECam) (DePoy et al., 2008) with a 30K × 30K FOV enable a single observer (or team of scientists) to generate over 1Tb of processed data in a single night, making all existing open source tools (other than *VisiOmatic*) inconvenient and inefficient.

To prevent users from loading an entire 2 Gb image at once, *Toyz* divides the image into a set of 400 × 200 px tiles and only loads the tiles into memory (and the browser) when they are visible in the viewing window. As the user scrolls horizontally and vertically more tiles are loaded on demand so that with a fast ethernet connection the user experiences almost no delay in loading images >2 Gb.

But in order to allow users to quickly load images of any size we sacrifice the quality of a subscaled image for faster processing. While we saw in Section 2.2 that python can load data into memory very quickly, it turns out that mapping an array of float values into RGB color tuples is very time intensive and mapping fewer points to our final tiles greatly improves performance. This means that instead of "properly" rescaling images using an interpolation or resampling function, we slice the data by taking every *n*th point, giving the tiles a slightly pixelated look. By using this method the images are loaded and displayed almost instantly on a local machine or when connected to a server with a very fast network connection. Most tiles range in size from 5–100 kB in size so for a standard 1366 × 768 px browser window with 16 tiles filling the viewer, the browser will need to load at most 1.6 MB, which takes just over a second with a 10 Mb/s connection.

Of course large FITS images are not the only types of images worth viewing in a browser. One of the byproducts of data analysis is often a large collection of plots that are generated wherever the data is stored and processed (in this case on a server). Depending on the software used, the filetype of these plots can vary and it is also useful to have the ability to view these plots without copying them from the server to the local machine. *Toyz* uses Pillow, a fork of the Python Imaging Library, which allows users to view a wide array of image formats including bmp, eps, jpeg, png, and tiff files, as well as *astropy* to load FITS images. Users are given the option as to whether an image is loaded as a mosaic of tiles or as a single image (which might be more useful in the case of small images like plots).

The image viewer consists of standard tools such as scaling, panning and centering, as well as a few additional handy features. Since many modern astronomical and scientific images contain multiple frames, a set of controls allows users to easily browse

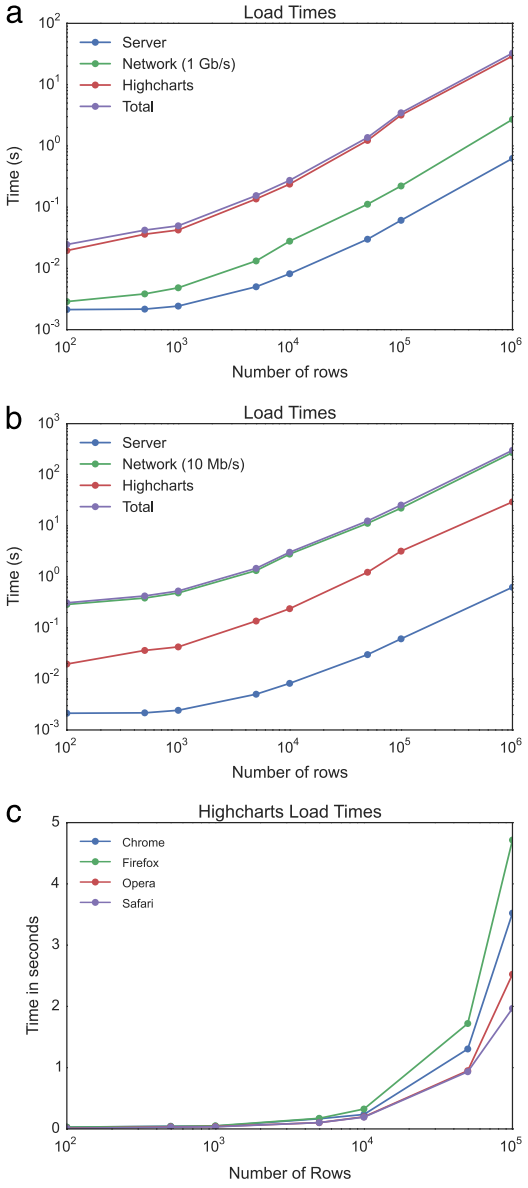


Fig. 4. Estimates of the time to load a datasets (based on the number of rows in the source) on a Macbook air connected to server running a Toyz instance with (a) a 1 Gb/s local network connection, (b) a 10 Mb/s internet connection. Using the same data sources, (c) compares the performance of the most popular browsers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

through the different frames of a single FITS image (see Fig. 3). It is also possible to have multiple viewer frames loaded at the same time, making it easy to switch or "blink" between different images (see Section 3.1 for more on blinking and other tools specifically related to astronomical FITS images).

The viewer is also designed to be customizable in that end users can add their own controls to the toolbar and even create their own custom image loaders. Since the viewer is also just a tile in a workspace, it is possible to have multiple viewers loaded at the same time and in the future it should even be possible to link the images to catalogs similar to *GLUE*.

2.4. Security

The typical *Toyz* install does not require much in the way of security. The recommended install of *Toyz* for research purposes is to install the application on a server, log on to the server using a secure shell and forward the port *Toyz* is running on from the server to the local machine. As long as the server is located behind a firewall the need for security is limited, especially if there is no reason for the users in the group to keep their analyses or data private from one another. In this simplest use case each user can be added to an admin group, allowing them access to all of the files on the server (that the account running the instance of *Toyz* has permission to access) and run any python module that conforms to the *Toyz* standard (see Fig. 6).

In other scenarios, for instance groups working with confidential data or classrooms where students should not have access to each others data or analyses, it is necessary for each user to have his/her own account. *Toyz* provides an admin console webpage that allows administrators to create new users and groups as well as change their permissions for a wide variety of features. By default, each user outside the admin group does not have permission to view any directories outside of the default directory created for him/her when his/her account is created and only administrators can change those permissions. This allows some directories to be shared by specified users or groups while remaining private from others.

More importantly, because *Toyz* acts as a GUI to the entire python library, without specific precautions taken a user would be able to run any python module, giving them the ability to run arbitrary code on the server. To combat this *Toyz* users and groups outside the admin group can only run python modules they are specifically given permission to run and no user is allowed to run a python module or function that does not conform to a specific standard given in the *Toyz* documentation.

3. Affiliated packages and extensions

A template is included with the *Toyz* source code that allows users to create their own custom *Toyz*. This allows them to create custom web pages, new workspace tile types, as well as python data types, classes, and functions for any purpose that their group sees fit, as long as they conform to the standards specified in the template. Many of the built-in *Toyz* functions can also be wrapped, as in the example of the Astro-Viewer (see Section 3.1), where even the control panel of the viewer has been created in such a way as to allow users to add their own buttons and controls. The *Toyz* website has a section for affiliated *Toyz* called the Toy Box which will host links to packages created by other users or groups built on the *Toyz* framework.

3.1. Astro-Toyz

To demonstrate the flexibility of *Toyz* as well as support our own research we developed an affiliated package called *Astro-Toyz*. While *Toyz* was designed for general data visualization and analysis, *Astro-Toyz* is designed specifically for the analysis of astronomical data. It contains add-ons to the image viewer that displays *world* coordinates, plot histograms or surface plots (similar to imexam in IRAF Tody, 1986), and align images in separate viewer frames to the same coordinates and scaling so that images can be blinked (see Fig. 5).

In the long run the goal is for *Astro-Toyz* to be a front end for *astropy*, providing a GUI for users to make use of *astropy* tools and affiliated packages such as converting WCS coordinates, object detection, matching to source catalogs from online sources like Vizier (Ochsenbein et al., 2000) by using *astroquery*, performing precision astrometry and photometry, and various other tasks supported in the *astropy* universe. We hope to entice the large community of developers who maintain other *astropy* packages to add their own interfaces onto *Astro-Toyz* to broaden its scope. At that time *Astro-Toyz* will be capable of being implemented in

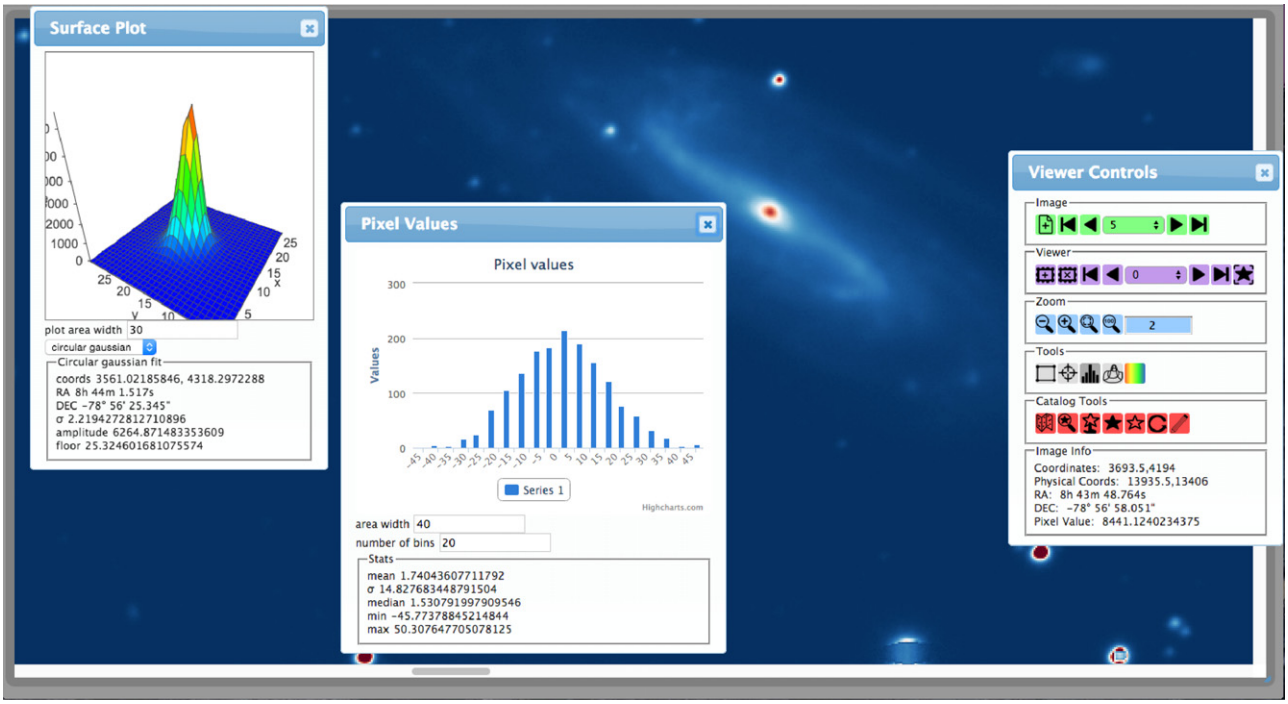


Fig. 5. Image displayed using the *Astro-Toyz* image viewer. DECam image in the vicinity of the spiral galaxy ESO 18–13. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

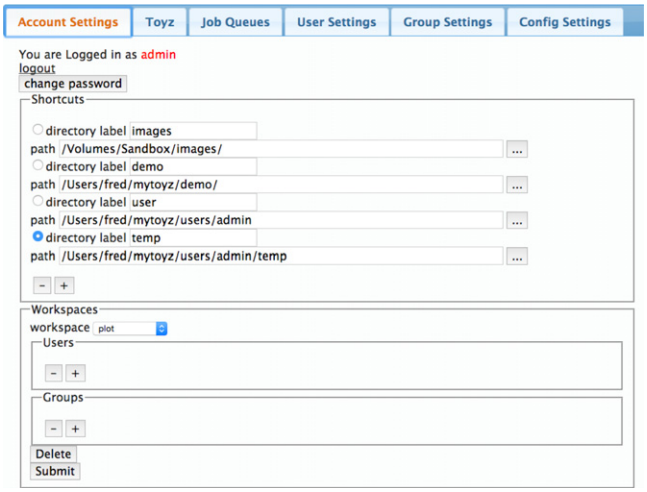


Fig. 6. Management console. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

undergraduate astronomy classes, allowing users to perform all of their analysis from their own computers without installing any software.

Currently *Astro-Toyz* is not quite ready for that level of interaction but does provide additional tools to the *Toyz* image viewer that give users access to a number of advanced features including displaying WCS and header info, access to the full matplotlib (Hunter, 2007) catalog of color maps, WCS alignment between images and blinking between multiple aligned images for moving object detection.

3.2. Extensions

At times it may also be useful to write a short module to extend the functionality of *Toyz* (or a *Toyz* affiliated package) for a specialized task. Fig. 7 shows an example of a custom workspace tile that loaded images and source information for point sources selected in one of the high charts files, used to track down saturated stars and other artifacts passing as point sources in our catalog. This tile was very specific to our observations and analysis and is not useful enough to make its own affiliated package, but it demonstrates the power *Toyz* gives its users to generate custom interactive content.

4. Future work

Toyz is still in its infancy and a number of exciting improvements are planned for the future. The biggest upgrade will be integration with Jupyter. Both iPython and Jupyter have similar APIs that allow users to run python code from a web browser but currently the interface to Jupyter is the traditional notebook format. We are in the process of designing a notebook extension that will implement the *Toyz* workspace interface in iPython, which will be useful for a wider community of users outside astronomy as well as making it easier for end-users to develop their own custom tiles and tools without extensive knowledge of javascript.

To improve collaboration an upgrade to workspaces is planned to allow multiple sessions to share the same set of session variables and client interfaces. This will make it possible for a user to open multiple tabs in a client browser that all connect to the same instance of a data source, saving memory and allowing changes to the data to propagate to all of the clients. Another advantage of this upgrade is that multiple users will be able to connect to the same workspace and get live updates as the workspace tiles are modified by their collaborators.

On the data visualization front there are plans to fully integrate *Toyz* with *GLUE* to give users access to the entire *GLUE* API in a web browser, allowing them to connect to both local and remote data. We have also been in contact with the *Ginga* collaboration to discuss integrating the current *Toyz* "slippy map" viewer with the extensive toolset developed by *Ginga* to create a more complete image viewing platform.

Due to limitations in browser memory *Highcharts* is limited in the number of data points that can be displayed at once in

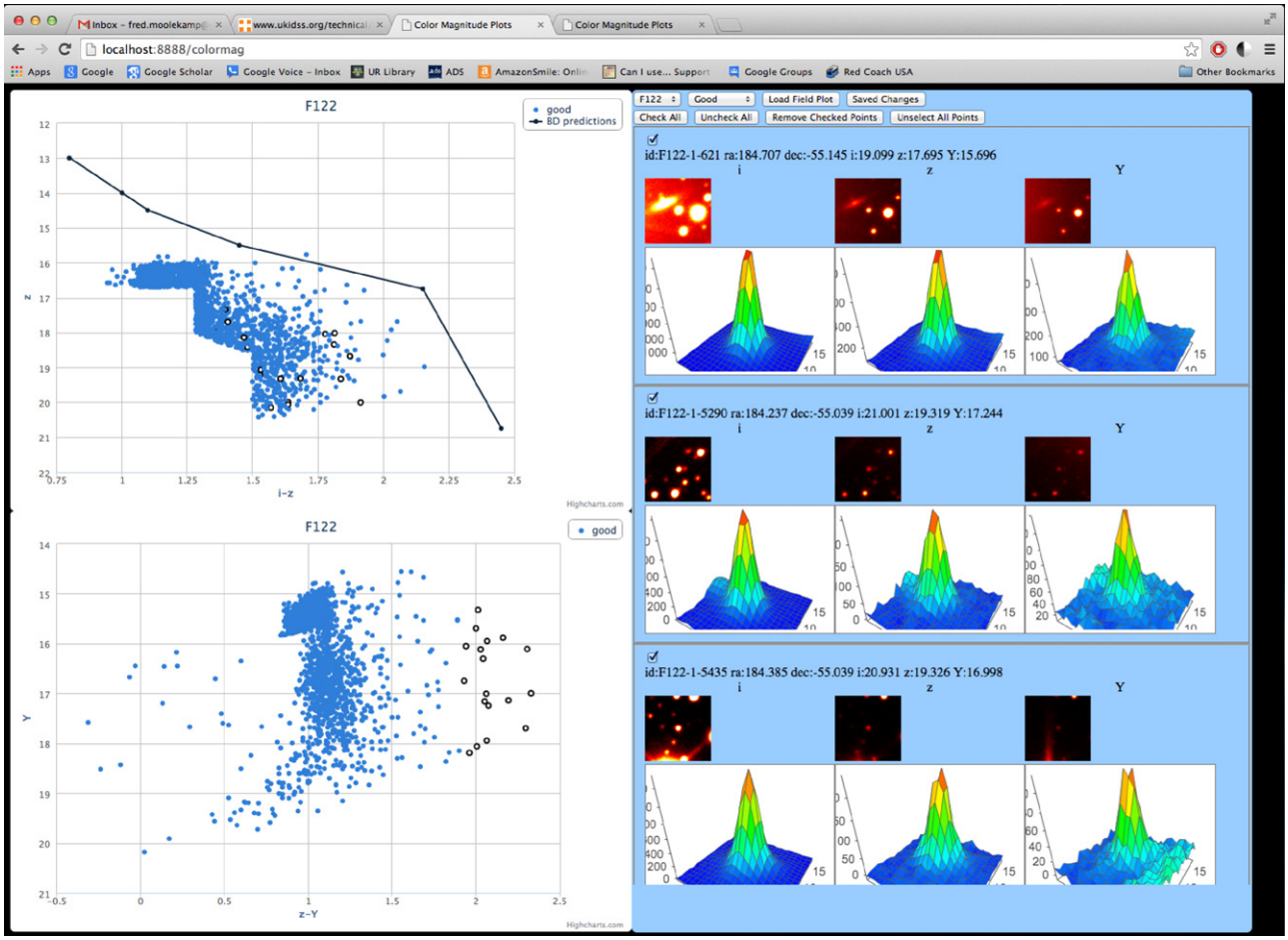


Fig. 7. Custom webpage to interact with a point source catalog. On the left are interactive *Highcharts* plots. On the right is a custom tile created to display an image and surface plot from images taken of the same field with three different filters. Each set of images on the right corresponds to a different source selected in the *Highcharts* plots and can be removed if the detection is an artifact. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

an efficient manner. More advanced technologies like WebGL are better suited for the task of displaying large datasets as they work off of browser plugins and expand the memory and functional capacities of web browsers.

As for *Astro-Toyz*, the biggest upgrade will be implementing a set of catalog tools, allowing users to create and plot a source catalog over a FITS image and add or remove sources. Much of this functionality has already been created but several recent upgrades to the way *Toyz* handles data sources have made it necessary to re-write much of the *Astro-Toyz* catalog backend. Once this is completed catalogs will be loaded as a data source in the same manner as datasets. In addition, we will continually be adding new features to incorporate more *astropy* functionality so that it can become a fully operational front-end to the most useful *astropy* functions, allowing undergraduates to process data in classes with little to no programming background.

The source code and documentation for *Toyz* is located at https://github.com/fred3m/toyz while *Astro-Toyz* can be downloaded from https://github.com/fred3m/astro-toyz where bug fixes or and new pull requests are always welcome.

Acknowledgments

We would like to thank the *astropy* community for leading the way in development of open source astronomical packages, NSF grant AST-1313029 for supporting our research, and Cameron Bell for proofreading and providing helpful comments in the preparation of this paper. We would also like to thank Klaus Lang for his assistance in making the *Toyz* and *Astro-Toyz* installation process much simpler, and to the RocPy meetup for many helpful suggestions and feedback, especially Rich Sarkis, who was essential in the early design phase of *Toyz* by providing insight into the best libraries, modules, and technologies to use.

References

Astropy Collaboration, Robitaille, T.P., Tollerud, E.J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A.M., Kerzendorf, W.E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M.M., Nair, P.H., Unther, H.M., Deil, C., Woillez, J., Conseil, S., Kramer, R., Turner, J.E.H., Singer, L., Fox, R., Weaver, B.A., Zabalza, V., Edwards, Z.I., Azalee Bostroem, K., Burke, D.J., Casey, A.R., Crawford, S.M., Dencheva, N., Ely, J., Jenness, T., Labrie, K., Lim, P.L., Pierfederici, F., Pontzen, A., Ptak, A., Refsdal, B., Servillat, M., Streicher, O., 2013. Astropy: A community Python package for astronomy. Astron. Astrophys. 558, A33. arXiv:1307.6212. doi:10.1051/0004-6361/201322068.

Beaumont, C., Robitaille, T., Borkin, M., 2014. Glue: Linked data visualizations across multiple files. Astrophys. Source Code Libr.

Bertin, E., Pillay, R., Marmo, C., 2015. Web-based visualization of very large scientific astronomy imagery. Astron. Comput. 10 (0), 43–53. URL: http://www.sciencedirect.com/science/article/pii/S2213133714000730. arXiv:1403.6025. doi:10.1016/j.ascom.2014.12.006.

Darnell, B., 2015. Tornado Developers, Tornado web server. https://github.com/tornadoweb/tornado.

DePoy, D.L., Abbott, T., Annis, J., Antonik, M., Barceló, M., Bernstein, R., Bigelow, B., Brooks, D., Buckley-Geer, E., Campa, J., Cardiel, L., Castander, F., Castilla, J., Cease, H., Chappa, S., Dede, E., Derylo, G., Diehl, H.T., Doel, P., DeVicente, J., Estrada, J., Finley, D., Flaugher, B., Gaztanaga, E., Gerdes, D., Gladders, M., Guarino, V., Gutierrez, G., Hamilton, J., Haney, M., Holland, S., Honscheid, K., Huffman, D., Karliner, I., Kau, D., Kent, S., Kozlovsky, M., Kubik, D., Kuehn, K., Kuhlmann, S., Kuk, K., Leger, F., Lin, H., Martinez, G., Martinez, M., Merritt, W., Mohr, J., Moore, P., Moore, T., Nord, B., Ogando, R., Olsen, J., Onal, B., Peoples, J., Qian, T., Roe,

N., Sanchez, E., Scarpine, V., Schmidt, R., Schmitt, R., Schubnell, M., Schultz, K., Selen, M., Shaw, T., Simaitis, V., Slaughter, J., Smith, C., Spinka, H., Stefanik, A., Stuermer, W., Talaga, R., Tarle, G., Thaler, J., Tucker, D., Walker, A., Worswick, S., Zhao, A., 2008. The dark energy camera (decam). URL: doi:10.1117/12.789466.

Federl, P., Grimstrup, A., Kiddle, C., Taylor, A.R., Robinson, K., Stephure, M., Yee, G., 2012. Remote visualization of large multi-dimensional radio astronomy data sets. In: Ballester, P., Egret, D., Lorente, N.P.F. (Eds.), Astronomical Data Analysis Software and Systems XXI. In: Astronomical Society of the Pacific Conference Series, vol. 461. p. 95.

Federl, P., Grimstrup, C.A.K., Taylor, A.R., 2011. On-line access and visualization of multi-dimensional FITS data. In: Evans, I.N., Accomazzi, A., Mink, D.J., Rots, A.H. (Eds.), Astronomical Data Analysis Software and Systems XX. In: Astronomical Society of the Pacific Conference Series, vol. 442. p. 467.

Fitzpatrick, M.J., Olsen, K., Economou, F., Stobie, E.B., Beers, T.C., Dickinson, M., Norris, P., Saha, A., Seaman, R., Silva, D.R., Swaters, R.A., Thomas, B., Valdes, F., 2014. The NOAO data laboratory: a conceptual overview. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 9149. p. 1. doi:10.1117/12.2057445.

Flaugher, B.L., Abbott, T.M.C., Angstadt, R., Annis, J., Antonik, M.L., Bailey, J., Ballester, O., Bernstein, J.P., Bernstein, R.A., Bonati, M., Bremer, G., Briones, J., Brooks, D., Buckley-Geer, E.J., Campa, J., Cardiel-Sas, L., Castander, F., Castilla, J., Cease, H., Chappa, S., Chi, E.C., da Costa, L., DePoy, D.L., Derylo, G., de Vincente, J., Diehl, H.T., Doel, P., Estrada, J., Eiting, J., Elliott, A.E., Finley, D.A., Flores, R., Frieman, J., Gaztanaga, E., Gerdes, D., Gladders, M., Guarino, V., Gutierrez, G., Grudzinski, J., Hanlon, B., Hao, J., Holland, S., Honscheid, K., Huffman, D., Jackson, C., Jonas, M., Karliner, I., Kau, D., Kent, S., Kozlovsky, M., Krempetz, K., Krider, J., Kubik, D., Kuehn, K., Kuhlmann, S.E., Kuk, K., Lahav, O., Langellier, N., Lathrop, A., Lewis, P.M., Lin, H., Lorenzon, W., Martinez, G., McKay, T., Merritt, W., Meyer, M., Miquel, R., Morgan, J., Moore, P., Moore, T., Neilsen, E., Nord, B., Ogando, R., Olson, J., Patton, K., Peoples, J., Plazas, A., Qian, T., Roe, N., Roodman, A., Rossetto, B., Sanchez, E., Soares-Santos, M., Scarpine, V., Schalk, T., Schindler, R., Schmidt, R., Schmitt, R., Schubnell, M., Schultz, K., Selen, M., Serrano, S., Shaw, T., Simaitis, V., Slaughter, J., Smith, R.C., Spinka, H., Stefanik, A., Stuermer, W., Sypniewski, A., Talaga, R., Tarle, G., Thaler, J., Tucker, D., Walker, A.R., Weaverdyck, C., Wester, W., Woods, R.J., Worswick, S., Zhao, A., 2012. Status of the dark energy survey camera (DECam) project. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 8446. p. 11. doi:10.1117/12.856609.

Foreman-Mackey, D., Hogg, D.W., Lang, D., Goodman, J., 2013. emcee: The MCMC Hammer. Publications of the ASP 125, pp. 306–312. arXiv:1202.3665. doi:10.1086/670067.

Gopu, A., Hayashi, S., Young, M.D., Harbeck, D.R., Boroson, T., Liu, W., Kotulla, R., Shaw, R., Henschel, R., Rajagopal, J., Stobie, E., Knezek, P., Martin, R.P., Archbold, K., 2014. Odi—portal, pipeline, and archive (odi-ppa): a web-based astronomical compute archive, visualization, and analysis service. URL: doi:10.1117/12.2057123.

Hickson, I., 2011. The websocket api. W3C Working Draft WD-websockets-20110929, September.

Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Comput. Sci. Eng. 9 (3), 90–95. URL: http://scitation.aip.org/content/aip/journal/cise/9/3/10.1109/MCSE.2007.55. doi:10.1109/MCSE.2007.55.

Jacoby, G.H., Tonry, J.L., Burke, B.E., Claver, C.F., Starr, B.M., Saha, A., Luppino, G.A., Harmer, C.F.W., 2002. Wiyn one degree imager (odi). URL: doi:10.1117/12.457017.

Jones, E., Oliphant, T., Peterson, P., et al. 2001. SciPy: Open source scientific tools for Python. URL: http://www.scipy.org/ [Online; accessed 19.06.15].

Joye, W.A., Mandel, E., 2003. New features of SAOImage DS9. In: Payne, H.E., Jedrzejewski, R.I., Hook, R.N. (Eds.), Astronomical Data Analysis Software and Systems XII. In: Astronomical Society of the Pacific Conference Series, vol. 295. p. 489.

Lins, L., Klosowski, J., Scheidegger, C., 2013. Nanocubes for real-time exploration of spatiotemporal datasets. IEEE Trans. Vis. Comput. Graphics 19 (12), 2456–2465. doi:10.1109/TVCG.2013.179.

Loebman, S., Ortiz, J., Choo, L.L., Orr, L., Anderson, L., Halperin, D., Balazinska, M., Quinn, T., Governato, F., 2014. Big-data management use-case: A cloud service for creating and analyzing galactic merger trees. In: Proceedings of Workshop on Data Analytics in the Cloud. DanaC'14. ACM, New York, NY, USA, pp. 9:1–9:4. URL: http://doi.acm.org/10.1145/2627770.2627774. doi:10.1145/2627770.2627774.

McKinney, W., 2010. Data structures for statistical computing in python. In: van der Walt, S., Millman, J. (Eds.), Proceedings of the 9th Python in Science Conference. pp. 51–56.

Merenyi, E., 2014. Hyperspectral image analysis in planetary science and astronomy. In: American Astronomical Society Meeting Abstracts 223. In: American Astronomical Society Meeting Abstracts, vol. 223. p. 337.

Ochsenbein, F., Bauer, P., Marcout, J., 2000. The vizier database of astronomical catalogues. Astron. Astrophys. Suppl. Ser. 143 (1), 23–32. URL: doi:10.1051/aas:2000169. arXiv:astro-ph/0002122.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: Machine learning in python. J. Mach. Learn. Res. 12, 2825–2830. arXiv:1201.0490.

Pillay, R., 2014. IIPImage: Large-image visualization. Astrophys. Source Code Libr.

Pogge, R.W., DePoy, D.L., Atwood, B., O'Brien, T.P., Byard, P.L., Martini, P., Stephens, A.W., Gatley, I., Merrill, M., Vrba, F.J., Henden, A.A., 1998. Mdm/ohio state/aladdin infrared camera (mosaic). URL: doi:10.1117/12.317266.

Ragan-Kelley, M., Perez, F., Granger, B., Kluyver, T., Ivanov, P., Frederic, J., Bussonier, M., 2014. The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication. AGU Fall Meeting Abstracts, D7, December.

Tody, D., 1986. The iraf data reduction and analysis system. URL: doi:10.1117/12.968154.

Valdes, F., Gruendl, R., 2014. The DECam community pipeline. In: Manset, N., Forshay, P. (Eds.), Astronomical Data Analysis Software and Systems XXIII. In: Astronomical Society of the Pacific Conference Series, vol. 485. p. 379. DES Project.

van der Walt, S., Colbert, S.C., Varoquaux, G., 2011. The numpy array: A structure for efficient numerical computation. Comput. Sci. Eng. 13 (2), 22–30. URL: http://scitation.aip.org/content/aip/journal/cise/13/2/10.1109/MCSE.2011.37. arXiv:1102.1523. doi:10.1109/MCSE.2011.37.

van der Walt, S., Schönberger, J., Nunez-Iglesias, J., Boulogne, F., Warner, J., Yager, N., Gouillart, E., Yu, T., 2014. the scikit-image contributors, scikit-image: image processing in python. PeerJ 2:e453. arXiv:1407.6245. doi:10.7717/peerj.453.

Wickham, H., 2009. ggplot2: Elegant Graphics for Data Analysis. Springer, New York, URL: http://had.co.nz/ggplot2/book. doi:10.1111/j.1541-0420.2011.01616.x.