DEVELOPING INTERACTIVE WEB APPLICATIONS FOR MANAGEMENT OF ASTRONOMY DATA

By

Dan Burger

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

May, 2013

Nashville, Tennessee

Approved:

Professor Keivan G. Stassun

Associate Professor William H. Robinson

To Naomi, Arnold, Anat, Oded and Sarah

ACKNOWLEDGEMENTS

This year marks my seventeenth year of affiliation with Vanderbilt University: two years as a graduate student, one year as a staff member, four years as an undergraduate student, and ten years of taking piano lessons at the Blair School of Music. I am thankful that Vanderbilt has continued to support me after all these years. I am grateful for all the experiences that I have had here, the professors that have guided me and the friends that I have made along the way.

Four years ago, Prof. Keivan Stassun gave me a chance to work on a project for a summer research program for undergraduates at Vanderbilt. He has supported me ever since, and I would like to thank him in particular for his kind and generous support. I would like to acknowledge the various members of the Stassun research group who have guided my research during this time and helped me to organize and revise the ideas in my thesis, specifically Dr. William Robinson, Dr. Joshua Pepper, Dr. Martin Paegert, Dr. Nathan DeLee, and Rob Siverd.

I would also like to acknowledge the various users of Filtergraph and the KELT voting system who have provided valuable feedback, as well as financial support from a NASA ADAP grant and from the Vanderbilt Initiative in Data-intensive Astrophysics (VIDA).

Finally, I would like to acknowledge my family: my parents, Naomi and Arnold Burger, my sister, Anat Burger, my brother, Oded Burger, and my sister-in-law, Sarah Greenberg. Your support and encouragement means a lot to me.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

There is a major shift in astronomy research from direct observation by telescopes to massive computerization of data. Most of it is driven by Moore's Law, the adage laid out by Intel's founder Gordon Moore that states that the processing power of computers doubling between 18 months and two years. As a result, modern telescopes are now able to capture large amounts of data. Such examples of this are the Kepler telescope, the Large Scale Survey Telescope, Atacama Observatory, and the Square Kilometer Array; the latter will produce one billion gigabytes of data per day when fully operational. [1] [2] [3]

With the large amounts of data produced comes the challenge of managing the data. Several projects managed by Zooniverse distribute the raw data among Internet users who volunteer to sift through the data in their spare time, a process known as crowdsourcing. This model has been adapted to fields both in and out of astronomy. [1] In addition, the Virtual Observatory is a worldwide consortium that makes raw data available to the astronomers who are able to manage the data. [4]

This research has been borne out of research on transiting exoplanets. Unlike exoplanets discovered through the radial velocity approach, transiting exoplanets are particularly useful because we can obtain much more information about them, such as its dimensions, temperature and atmosphere. [5] Two approaches for managing large-scale astronomy projects are described here. One approach uses small-scale crowdsourcing to manage data. The other allows anyone to build interactive data visualization portals from an uploaded set of collected data. These two approaches were developed using the Python programming language and the Web2py web framework.

CHAPTER II

A VOTING SYSTEM TO SELECT EXOPLANET CANDIDATES FOR THE KELT TRANSIT SURVEY

Introduction

The KELT (Kilodegree Extremely Little Telescope) project is a long-term photometric survey of a large fraction of the sky, focusing on stars with $8 < V < 10$. The KELT survey is measuring the brightness of these stars to discover new exoplanets by identifying the periodic dimming of the stars caused by the transits of planets whose orbits are aligned with our line of sight.

KELT consists of two near-identical robotic telescopes – KELT-North is operated by the Ohio State University and is located at Winer Observatory in Sonoita, AZ [6], while KELT-South is operated by Vanderbilt University and is located at the South Africa Astronomical Observatory in Sutherland, South Africa [7]. The KELT science team consists of astronomers located at those two institutions, as well as other institutions across the world.

The KELT-North and KELT-South telescopes operate through pre-programmed observing scripts, gathering their data by repeatedly observing a set of predefined fields located across the sky, and they transfer the raw data and calibration files back to Ohio State and Vanderbilt, respectively. The data for each field on the sky are separated out, and the raw data are calibrated and reduced, field by field. All stars in the fields observed are identified, their brightnesses measured, and the resulting plots of each star's brightness over time, called the star's light curve, is saved.

A series of filters are then applied, based on information about each star that has been observed, and only stars with suitable properties that will allow exoplanets to be detected are selected. To the light curves of those stars we then apply an algorithm called box least square (BLS) which searches each light curve for signs of small periodic dips in brightness indicative of a transiting planet. That process produces, for each field, a ranked list of light curves according to the strength of the signal uncovered by BLS, above some predefined cut in signal strength. Additional algorithms attempt to screen out giant stars and to select stars within a suitable range of temperatures. For each field analyzed, the final list of stars that pass that cut can be dozens to hundreds. The stars in that list are referred to as “candidates”.

The Problem

At this point, we have exhausted the available tools for automatically selecting promising candidates for transiting exoplanets. The reason is that there are a large number of both instrumental systematics and astrophysical phenomena that can mimic the signal of a transiting exoplanet in a star’s light curve. Further astronomical observations are required to confirm whether a candidate is in fact a transiting exoplanet. Those observations require time and effort, and we therefore need a method for determining the likelihood that each of the candidates is worth this additional investment.

Separating possible exoplanet signatures from the various false positives requires a holistic evaluation of the properties of the candidate star, its light curve, and associated data. No existing algorithm can make that evaluation, and we therefore require a mechanism to allow members of the KELT team to make their own evaluations and judge the status of each star, and to then assemble those separate judgments into a collective evaluation for the status of each star. Those collective evaluations are then used as the basis for group discussions to narrow the list of hundreds of candidates per field to 2-3 dozen that are then sent for further observations, which we call “selected candidates”. The evaluations are based on an internal project website that assembles a wide range of information and plots for each candidate.

Any solution for managing the evaluations of each team member needs to meet various constraints. Since team members are spread out geographically, the voting system must be designed for remote access among various operating systems. Furthermore, the voting system must be integrated into the project’s workflow environment so that as team members make an evaluation for each candidate, they can easily sift through the vast amounts of data collected for that candidate.

One solution would be to have each team member communicate their evaluations by e-mail. While it is simple to implement initially, it results in more work for the team members in sending their evaluations as well as the administrator for tabulating the results. Another solution would be to use an "off-the-shelf" cloud-based polling service such as SurveyMonkey [8] or the forms feature in Google Drive [9]. However, neither of these services can provide the flexibility or features needed for our project.

In order for us to handle candidate evaluations smoothly, a custom web-based voting system is needed. In this chapter we discuss a voting system for helping KELT team members accomplish their goals.

Function of the Voting System

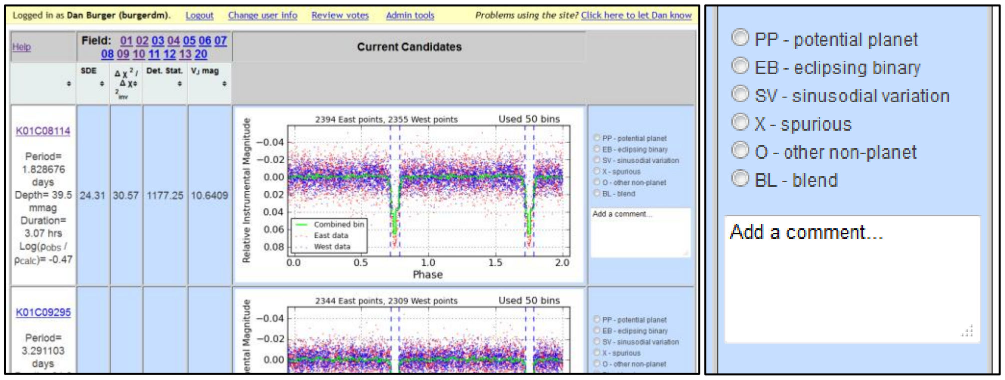
Access to the KELT voting pages is limited to KELT team members. Registered users of the site have access to some basic administrative tools such as changing their user info and obtaining lost passwords. Regular users are able vote for candidates and provide comments. There are two special types of users. Administrative users have access to a section of the site where they can create new voting sessions, stop a voting session in progress, add and delete other users, and modify information about another user. Non-voting users can only view the results of a voting session. There is also a guest user account category that only displays voting results.

Each field of KELT data is reduced separately, and contains between 60,000 and 140,000 stars with extracted light curves. After all the automated cuts, there are between approximately 50 and 500 candidates in need of individual evaluation by team members. A site administrator creates the candidate pages that display all the relevant candidate information, combining the light curves and data from the KELT survey with matched information from a variety of standard astronomical catalogs, as well as public time-series photometry from the SuperWASP [10] and ASAS [11] surveys. Each candidate has five associated pages: a main data page; a linked page showing various time series variability statistics and plots; a linked page showing the associated SuperWASP data phased to various periods; a linked page showing the associated ASAS data phased to various periods; and finally a page showing stacked image stamps comparing in-transit vs. out-of-transit images, which we call the residual flux plot (RFP) page. All the candidate pages are compiled in an index page showing very short data summaries and a single lightcurve plot for each candidate, all on the same page.

The administrator sets the starting time and ending time of each voting session. During the time period that voting is open, users can cast a vote for each candidate and add comments with each vote. The choices for each candidate are: Potential Planet (PP), Eclipsing Binary (EB), Sinusoidal Variation (SV), Spurious (X), Other non-planet (O), and Blend (B). Candidate votes and comments can be revised at any time while voting is open for that field, although each user can only view their own votes and comments until the voting session for that field is closed.

Comments are useful for allowing voters to take notes on each candidate, and for pointing out especially relevant pieces of information on the candidate pages. There have been a few cases where a majority of users cast votes for one option but discussions in the comments made it clear that the candidate belonged in a different category

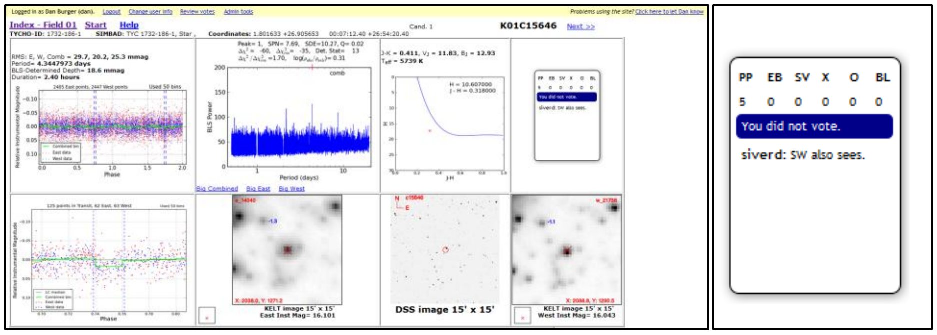
Once the voting is closed, all results and comments are visible to all users. Users do not see what votes each other user made, only the compiled totals, but they can view the username of those who added comments.


Logged in as **Dan Burger (burgerdm)**. Logout Change user info Review votes Admin tools *Problems using the site?* Click here to let Dan know

Help	Field: 01 02 03 04 05 06 07 08 09 10 11 12 13 20				Current Candidates	
	SDE	$\Delta\chi^2 / \Delta\chi^2_{inv}$	Det. Stat.	V_J mag		
K01C08114 Period= 1.828676 days Depth= 39.5 mmag Duration= 3.07 hrs Log(ρobs / ρcalc)= -0.47	24.31	30.57	1177.25	10.6409		PP - potential planet EB - eclipsing binary SV - sinusodial variation X - spurious O - other non-planet BL - blend Add a comment...
K01C09295 Period= 3.291103 days						PP - potential planet EB - eclipsing binary SV - sinusodial variation X - spurious O - other non-planet

PP - potential planet
EB - eclipsing binary
SV - sinusodial variation
X - spurious
O - other non-planet
BL - blend
Add a comment...

Figure 1a-b. The index page for a KELT field where voting is in progress (1a, left). Figure 1b (right) shows a close-up of the voting options and comment field provided for each candidate.


Logged in as Dan Burger (dan). Logout Change user info Review votes Admin tools

Problems using the site? Click here to let Dan know

Index - Field 01 Start Help Cand. 1 **K01C15646** Next >>

TYCHO-ID: 1732-186-1 **SIMBAD:** TYC 1732-186-1, Star , **Coordinates:** 1.801633 +26.905653 00:07:12.40 +26:54:20.40

RMS: E, W, Comb = **29.7, 20.2, 25.3 mmag**
Period= **4.3447973 days**
BLS-Determined Depth= **18.6 mmag**
Duration= **2.40 hours**

Peak= 1, SPN= 7.69, SDE=10.27, Q= 0.02

J-K = **0.411**, V_J = **11.83**, B_J = **12.93**
T_{eff} = **5739 K**

H = 10.607000
J - H = 0.318000

PP	EB	SV	X	O	BL
5	0	0	0	0	0

You did not vote.

siverd: SW also sees.

Relative Instrumental Magnitude
Phase
BLS Power
Period (days)
comb
J-H
Big Combined
Big East
Big West
DSS image 15' x 15'
KELT image 15' x 15' East Inst Mag= 16.101
KELT image 15' x 15' West Inst Mag= 16.043

Figure 2a-b. A page for a KELT candidate (2a, left). The voting for that candidate has closed. Figure 2b (right) shows a close-up of the voting results for that candidate.

Technical details

The KELT voting system was built using Web2py, a framework for developing web applications using Python. Voting data and user information is stored in a MySQL database (see Figure 3), with Web2py automatically generating tables for user data.

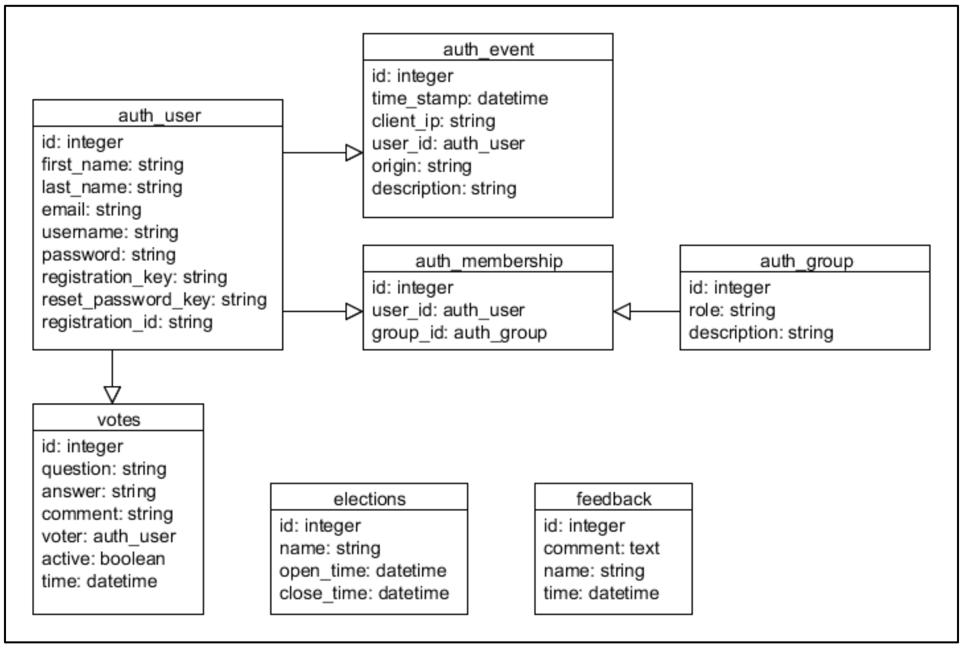


Figure 3. The database diagram for the KELT voting system. Tables starting with "auth" are automatically generated by Web2py. The tables "auth_permission" and "auth_cas" are also generated by Web2py but are not used.

When a user casts a vote in the system, the variable “question” contains the name of the candidate and the variable “answer” is set to the choice made by the user. If this is the first vote, then the active variable is set to true. If the vote gets changed, then the original vote is kept, but the active variable is set to false. The new vote would then have an active variable set to true. When tabulating the results, inactive votes are ignored as well as test votes cast using the author’s login.

Originally, all the KELT candidate pages were displayed in static files on a publicly accessible server, with voting ballots and results inserted into the page using an HTML iframe tag. However, each iframe tag conducts a request to the server. This means that a large index page with 268 candidates would perform 268 page load requests for candidate ballots or results; one for each candidate displayed. To alleviate this problem, a quick loading system was added to the HTML pages. The server would load the original index page file and modify the HTML code to add the voting ballots or results. The modified page is then returned to the user. Quick loading index pages also contain Javascript code that loads the images in a page when the user scrolls down, a technique known as lazy loading [12].

All other pages with voting ballots or results continue to use iframe tags; however, these pages are also modified by the server so that they can link to the newer quick voting pages. Handling those pages through the voting server also protects the data from unauthorized access. Iframe tags continue to be used for displaying the navigation bar at the top of the page.

Future work

One of the problems with the voting system is the need to make adaptations to the code every time the format of the original HTML files change, even if these changes are small. More work is needed to make the system more resilient to future updates to the original HTML files.

Although the quick loading pages are very specific to the KELT project, the HTML iframe widgets can be adapted to similar projects both in and out of astronomy. The advantages of doing this are to allow only registered users to vote, to allow users to add comments, and to set given times when users can vote and when users can view the results.

CHAPTER III

FILTERGRAPH: AN INTERACTIVE WEB APPLICATION FOR DATA VISUALIZATION OF ASTRONOMY DATASETS

Introduction

Increasingly in astronomy there is a need for performing quick-look inspection and visualization of large datasets in order to easily ascertain the nature and content of the data, begin identifying possibly meaningful structures or patterns in the data, and guide more computationally costly deep-dive analyses of the data. For example, consider that the first products of a large survey project is often a large database with many columns (representing the various measurable and/or derived quantities) and with many rows (representing the individual objects of study); it is not uncommon for such databases to include tens of columns and millions of rows.

To even begin visualizing such datasets---let alone perform basic analyses---can be a daunting task. The researcher is faced with questions such as: What is the content and what does it look like? Where are the "holes" in the data (missing or bad data) and are there systematics or biases of which to be wary? What are the relationships among the variables in the dataset, and are they meaningful? Are there interesting patterns that might be worthy of deeper investigation and analysis?

Indeed, the sheer size of such datasets and their multidimensionality is at the heart of what makes their visualization so challenging. To identify potentially meaningful patterns often requires "seeing" the data simultaneously across multiple dimensions and with appropriate "slices" through multiple multidimensional spaces.

Arguably even more fundamental to the visualization challenge is what might be called the high "potential barrier" that the user faces to even begin the visualization process. Certainly there exist high-end tools for storing data in a database, plotting data, and so on. But for many researchers there is from the first instance a very large overhead associated with using such tools: importing the data and correctly specifying meta-data, keeping track of a large number of variable names, issuing and scripting commands for plotting, plotting pairwise variables against one another over multiple iterations, attempting to filter out bad data, re-rendering plots to restricted data ranges, attempting to represent multiple variables at once, and so on. Faced with such high overhead in time and effort, there is the temptation to either skip the crucial quick-look visualization step altogether, or else to make very limited attempts at representing the data with simple plots based on preconceptions about what should be meaningful to visualize.

We have developed Filtergraph as “Plotting 2.0”, an easy-to-use web-based solution to this problem. The principal motivation for Filtergraph is to make the “activation energy” of beginning the data visualization process as small as possible. Users can register to use the tool instantly, can immediately upload datasets without the requirement of meta-data specification, and can thus begin seeing their data in seconds. We have also sought to make Filtergraph intuitive (see sample in Figure 4). Plots involving 2, 3, 4, or 5 dimensions (3D + color + symbol size), as well as histograms, can be generating with a single click, and variable name fields are pre-filled and auto-complete so that the user does not need to remember the full content of the dataset in order to make plots. Mathematical operations on individual variables---or indeed among variables---can be performed on the fly. (For example, one can simply specify the x-axis of a plot as bmag-vmag.

And, true to its name, data can be easily filtered, for example by specifying data ranges explicitly or by dragging over a region of interest on the screen, rotating a plot, and other similarly intuitive gestural commands.

Importantly, Filtergraph is fast. There is nothing more frustrating or deterrent of the creative visualization process than being faced with long lag times between subsequent plot renderings. If plots do not update instantly, users will be more likely to avoid the penalty associated with trial and error exploration. Filtergraph renders a plot of 1.5 million points in approximately one second, enabling and encouraging natural, seamless interaction with and exploration of the data. The user can change variables, attempt different mathematical operations, filter in and out, move back and forth between different representational forms, over and over, in seconds and without cognitive interruption.

Finally, Filtergraph is designed to facilitate sharing. Any Filtergraph plot can be saved as a graphics file in various formats. Filtered subsets of the data can also be saved as tables in various formats. More importantly, each dataset is instantly set up as a sharable ``data portal" (http://filtergraph.vanderbilt.edu/yourname) that can be provided to collaborators. Instead of sending collaborators a copy of the raw data file, the user can easily provide a simple URL that contains the data and the ability to instantly visualize it, thus greatly facilitating the collaboration process.

While there are data visualization services with similar interfaces, such as the Exoplanet Data Explorer [13] and Gapminder [14], the data on these services is fixed. Filtergraph is unique because it allows the user to visualize any given set of data.

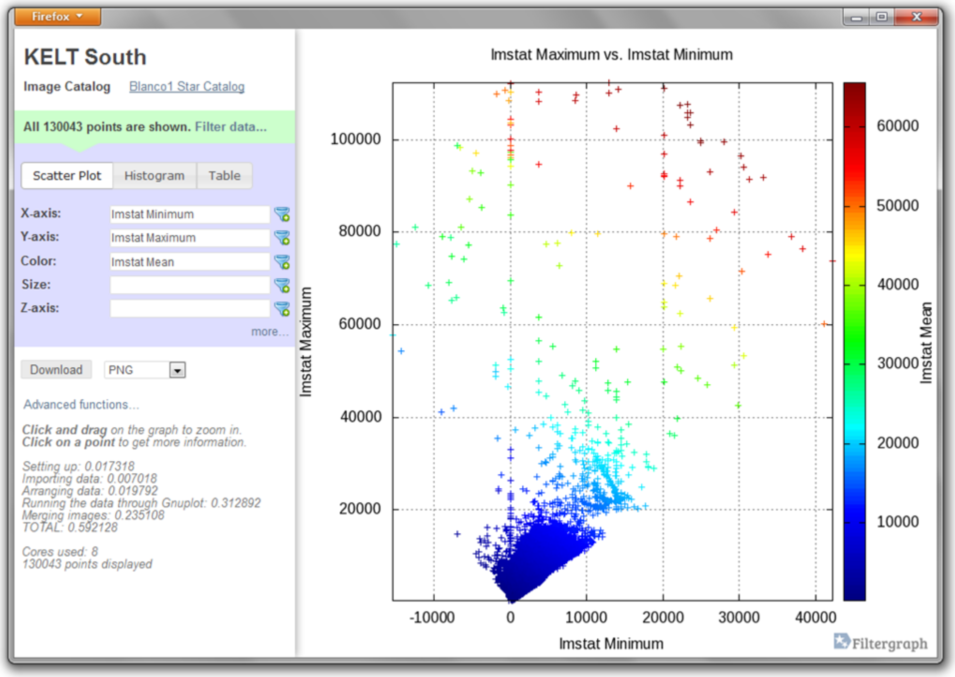

KELT South

Image Catalog   Blanco1 Star Catalog

All 130043 points are shown. Filter data...

X-axis: Imstat Minimum
Y-axis: Imstat Maximum
Color: Imstat Mean
Size:
Z-axis:

Click and drag on the graph to zoom in.
Click on a point to get more information.

Setting up: 0.017318
Importing data: 0.007018
Arranging data: 0.019792
Running the data through Gnuplot: 0.312892
Merging images: 0.235108
TOTAL: 0.592128

Cores used: 8
130043 points displayed

Figure 4. A screenshot of a Filtergraph portal generated from KELT South data. This portal is publically available at http://filtergraph.vanderbilt.edu/keltsouth/.

Current capabilities and use

Filtergraph is a free web-based service. By uploading a dataset to Filtergraph, the user is presented with a portal that can be used to generate plots and tables based on the dataset. The portal can be shared with others and provides interactive features such as zooming and obtaining information about a point on the scatter plot.

Filtergraph currently supports visualization in three forms. A scatter plot consists of at least two columns; one for the X-axis and one for the Y-axis (see Figure 4). Three optional axes may be set. The color axis sets the color of each point based on a range of colors; currently higher values are colored red, lower values are colored blue, and values in between are placed along the color spectrum. Similarly, the size axis sets the size of each point with higher values receiving larger points and lower values receiving smaller points. The Z-axis transforms the scatter plot into a three-dimensional plot which can be viewed from different angles.

In addition to the scatter plot mode, the data can be plotted as a one-dimensional histogram or two-dimensional heat map, each of which is split based on a set number of bins. The two-dimensional heat map can be plotted as a three-dimensional surface that can be viewed from different angles.

Finally, Filtergraph can also return the data as a table based on a selected subset of rows and columns in the dataset. This table can be sorted by a column in the table, ascending or descending. The output table can be easily shared with others using ASCII and HTML formats. This is particularly useful for target selection, for example.

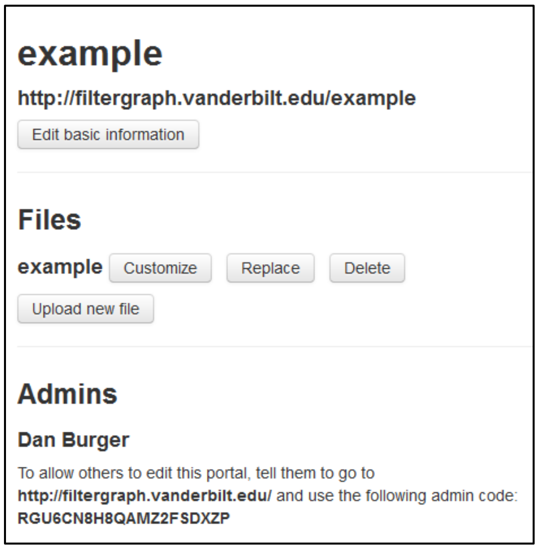
Filtergraph currently accepts a number of file formats for uploaded datasets. Plain text ASCII files are accepted as space separated, comma separated (CSV), tab separated (TSV), and fixed width formats. In these cases, Filtergraph will automatically read the file to determine several of its properties: which of these types are being uploaded, where the data begins in the file, whether or not there is a header in the file, the number of columns that should be imported, and the format types for each column. There is no need to include metadata in the file. In addition, Filtergraph accepts Microsoft Excel, SQLite, VOTable, FITS, IPAC, and Numpy file types. Upon upload, Filtergraph determines the structure of the dataset and populates all subsequent interfaces with the variable names determined from the header row.

A Filtergraph portal consists of two parts, with the left sidebar being used to control the main content (see Figure 4). Key components of the left sidebar include: the name and description of the portal; the ability to switch between datasets on the portal, if more than one dataset is available; the ability to apply criteria, or "filters", on the dataset; the ability to switch between the scatter plot, histogram, and table modes; the ability to apply display settings; the ability to output the data to a file (PNG, JPEG, GIF, Postscript, and PDF for graphs, HTML and ASCII for table data); and additional instructions and status info for using the dataset.

Axes are changed using an editable drop-down box that can include the name of an axis or advanced functions on one or more axes. The following advanced functions are supported: addition, subtraction, multiplication, division, modulo, power, natural logarithm, base 10 logarithm, absolute value, square root, exponential, pi, sine, cosine, tangent, and the hyperbolic and inverse versions of sine, cosine, and tangent.

For scatter plots and histograms, the main content can be modified interactively. Clicking on a point on the graph displays a popup window with all of the data for that particular point. Additionally, clicking and dragging on the window allows the user to zoom in on a particular region of the graph. A link is then provided to reset the zoom. Also, when the Z-axis is enabled for scatter plots, the user can rotate the 3-D scatter plot.

Administrative interface


example

http://filtergraph.vanderbilt.edu/example

Files

example

Admins

Dan Burger

To allow others to edit this portal, tell them to go to **http://filtergraph.vanderbilt.edu/** and use the following admin code: **RGU6CN8H8QAMZ2FSDXZP**

Figure 5. A screenshot of the administrative interface.

Filtergraph also contains an administration interface (see Figure 5) for maintaining multiple portals, each of which may contain one or more datasets. An administrator for a portal can change or delete the portal and add, change or delete datasets associated with the portal. An administrator can also select another user to co-administer the portal by sharing a random string of characters (an "admin code") that is associated with each portal.

Once a user registers for the site, he or she is asked to create a portal. To create a portal, the user provides a name and URL and then uploads an initial dataset. This dataset is then inspected to determine the data types for each column. At this point, the portal is ready for use; the user may optionally provide default settings for the dataset as well as alternate names for each of its columns. Filtergraph also provides many standard user administration features provided by the Web2py web framework, such as changing profile information and obtaining lost usernames and passwords.

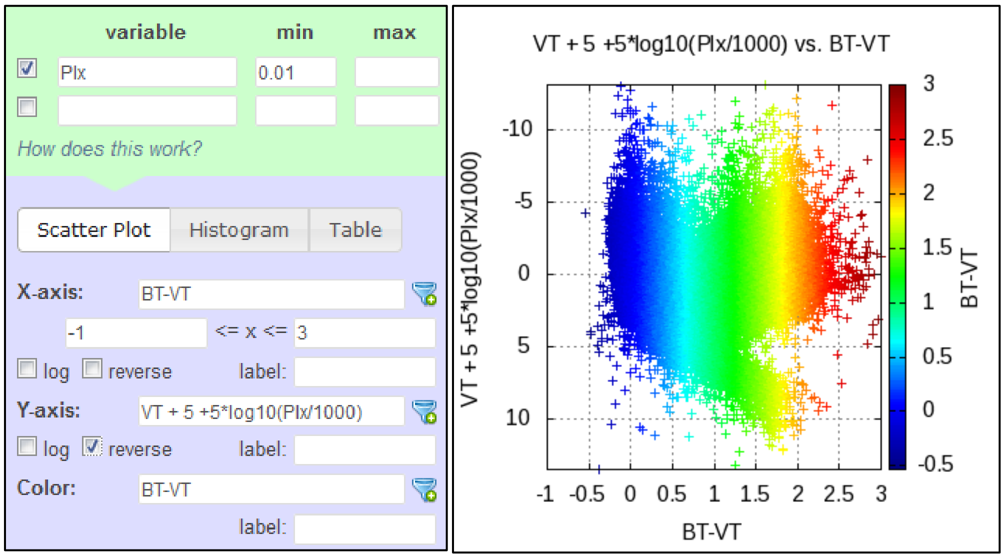
Customization

Some portals have been manually customized to display additional information on the popup window that appears when clicking on a point. For instance, the SLoWPoKES portal (http://filtergraph.vanderbilt.edu/slowpokes) displays information from the SDSS gri composite image and the 2MASS H-band based on data from the given point. This is accomplished by interpreting XML data from the NASA/IPAC Infrared Science Archive (http://irsa.ipac.caltech.edu). Similarly, another portal has been customized to generate a graph based on an IDL script. Thus, while the Filtergraph application is being provided with a set of standard features, its construction from a programming standpoint is sufficiently general to enable specific customization as needed.

Case studies

The original motivation for developing Filtergraph is to manage data coming from the KELT South telescope, which is a fully robotic telescope operated by Vanderbilt University and the South African Astronomical Observatory that searches for transiting exoplanets. KELT South generates a lot of images, and analyzing even one of these images takes a significant amount of time. By using the web portal for KELT South, we are able to select images for analysis effectively.

Filtergraph is finding broad use across many astronomical data visualization needs. Here we use the Hipparcos dataset to illustrate a few representative case studies. A sample portal for Hipparcos has been set up at http://filtergraph.vanderbilt.edu/hiptest. As a first example, we generate a Hertzsprung-Russell diagram (see Figure 6):



variable	min	max
✓ Plx	0.01	
☐		

How does this work?

Scatter Plot | Histogram | Table

X-axis: BT-VT
-1 <= x <= 3
☐ log ☐ reverse label:

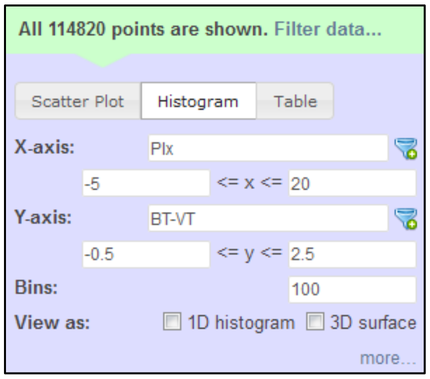
Y-axis: VT + 5 +5*log10(Plx/1000)
☐ log ✓ reverse label:

Color: BT-VT
label:

Figure 6a-b. By applying the settings on the left (Figure 6a) to the publically available Filtergraph portal at http://filtergraph.vanderbilt.edu/hiptest, the Hertzsprung-Russell diagram on the right (Figure 6b) can be generated.

First, we ask to screen out all points where Plx, the parallax variable, is less than 0.01. This is necessary so that all parallax data can be calculated using the logarithmic function presented in the Y-axis. For clarity, any outlier points where BT-VT is less than -1 or greater than 3 are also removed, where BT and VT are the B and T magnitudes from the Tycho catalog, respectively. The remaining points are plotted based on the functions given for the X axis, Y axis, and color axis. The separation between dwarf stars and giant stars is apparent.

Beyond two-dimensional scatter plots, Filtergraph can produce a much wider variety of images. As an example, Figure 7 depicts a two-dimensional histogram produced using Hipparcos data, with colors assigned to each square region of the image based on the density of data points in that region. In Figure 8, we use KELT-North data to generate a three-dimensional scatter plot with color as an additional axis. In the web interface, this three-dimensional display can be viewed from different angles using links at the bottom of the page.


All 114820 points are shown. Filter data...

Scatter Plot | Histogram | Table

X-axis: Plx
-5 <= x <= 20
Y-axis: BT-VT
-0.5 <= y <= 2.5
Bins: 100
View as: ☐ 1D histogram ☐ 3D surface

more...

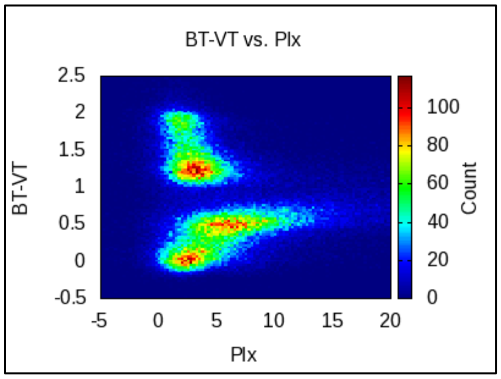
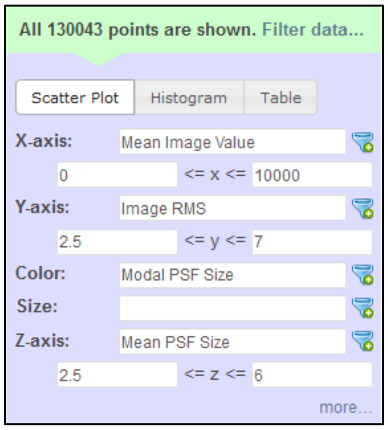


Figure 7a-b. By applying the settings on the left (Figure 7a) to the publically available Filtergraph portal at http://filtergraph.vanderbilt.edu/hiptest, the two-dimensional histogram on the right (Figure 7b) can be generated. The image was generated at a size of 400x300 pixels.


All 130043 points are shown. Filter data...

Scatter Plot | Histogram | Table

X-axis: Mean Image Value
0 <= x <= 10000
Y-axis: Image RMS
2.5 <= y <= 7
Color: Modal PSF Size
Size:
Z-axis: Mean PSF Size
2.5 <= z <= 6

more...

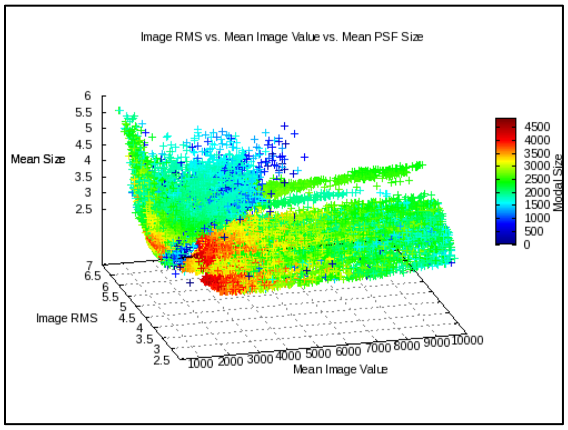


Figure 8a-b. By applying the settings on the left (Figure 8a) to the publically available Filtergraph portal at http://filtergraph.vanderbilt.edu/keltnorth, the three-dimensional scatter plot on the right (Figure 8b) can be generated. The following

advanced settings were applied: X-axis reversed, Y-axis reversed, Z angle set to 60 and 165 degrees, font size set to 8, image size set to 500x375 pixels, and color and z-axis labels set to the truncated "Modal Size" and "Mean Size", respectively.

Technical details of Filtergraph

Filtergraph was written in Python and developed using the Web2py framework. Web2py is a full-stack web framework that allows web applications to be written in Python and deployed easily. Web2py was chosen for a variety of reasons. It can be deployed easily on any Windows, Macintosh or Linux machine, and is compatible with Apache for public access to the web server. It also comes with a secure and comprehensive administrative interface that can be used to edit the code, upload files, examine database entries, and view errors that have occurred. [15]

A number of plugins and applications are used to support Filtergraph. On the server side, the Numpy library for Python is used to store the data efficiently and perform functions quickly on the data. [16] The server also invokes the Gnuplot application for producing graphs [17] and the Graphicsmagick application for performing additional image manipulation [18]. On the client side, the ImgAreaSelect library is used to allow users to zoom in on the graph. [19] The JQuery [20] and JQueryUI [21] libraries are also used to enhance the browser experience. Other third-party Python libraries used for processing data are ATpy [22], PyFITS [23], VO [24], and XLRD [25].

Filtergraph stores general information about each dataset in a MySQL database; the datasets themselves are stored in the file system. There are six tables in the database defined by Filtergraph, each of which store information about portals, datasets, administrators for each portal, headers contained in each dataset, and user feedback. There are also a few tables that are automatically generated by Web2py, the most important of which is "auth_user", which stores user information. For security purposes, passwords are encrypted in the system using the SHA-512 algorithm.

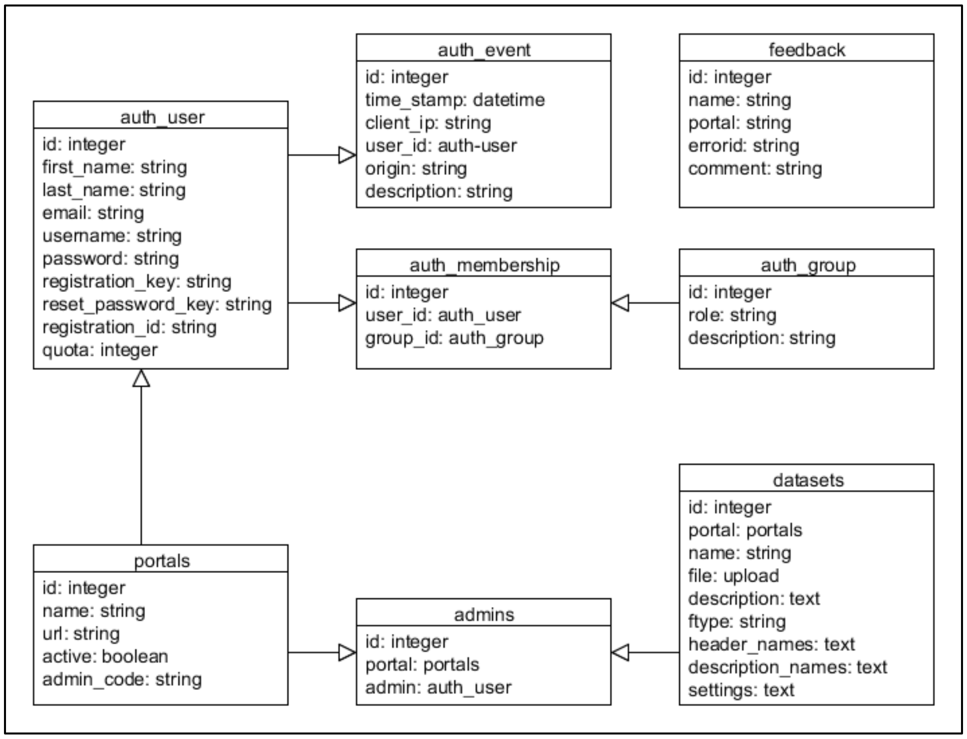


Figure 9. The database diagram for Filtergraph. Note that uploaded data is stored in the file system and not in the database. Tables and columns that are not currently in use have been omitted.

When a data file is uploaded to Filtergraph, the server performs an inspection on the file to determine the type of data each column represents. The headers are stored in the database (see Figure 9) with the name of the header and its type. Depending on the type, the headers may also include the minimum value, the maximum value, and its length (number of digits or characters). This information is used to store the data as efficiently as possible. Once the portal is accessed for the first time, the original data file is loaded and then stored as a binary file in a cache directory. Any subsequent loads would come from the cache file.

An important need for Filtergraph is to generate images quickly for very large datasets up to millions of rows. To optimize it for speed, Filtergraph uses an “embarrassingly parallel system” to process the graph based on the MapReduce paradigm. [26] Once the data is loaded and processed using Numpy, the instructions and binary data are distributed equally among one to *N* instances of

Gnuplot, where *N* is the number of cores in the system (eight at the time of this writing). This limit is imposed because running more than *N* processes at a time would not create a time advantage. Each instance of Gnuplot generates a PNG image containing its share of the data. The images are then merged together using Graphicsmagick and converted to the desired file format. The server returns the resulting image as well as information needed for the browser to support the zoom feature.

As the number of Gnuplot instances increases, it takes less time to generate the intermediate images but it takes more time to merge these images together. Filtergraph determines how many instances of Gnuplot to run by using equations to calculate how long it would take to generate the graph under one instance, under two instances and so on. The number of instances that would take the least time to generate the graph is then used. The equations are obtained using the Eureqa analytical software package based on the times of graphs produced by Filtergraph under varying conditions. [27] These equations are specific to our hardware setup, however in principle it should be possible to implement into Filtergraph the ability to determine these equations for any server setup, such as through a “benchmark” function to be added to the admin interface.

Since it is less practical to apply these procedures on Postscript and PDF files, generating these graphs always uses one process of Gnuplot. The same is true for histograms, since they typically do not require as much data to generate. For obtaining tables and point information, Gnuplot and Graphicsmagick are bypassed entirely and the resulting data is returned directly to the browser.

Future Plans

Filtergraph currently has over 100 users in over 20 countries. We would like to see its use expanded beyond astronomy to other academic and non-academic fields where data is being heavily used. We also plan to add more features to Filtergraph such as improved statistical capabilities and the ability for users to save the current settings of their portal for later use.

REFERENCES

[1] N. Sadasivam, "The astronomical data explosion," ScienceLine, 3 February 2013. [Online]. Available: http://scienceline.org/2013/02/the-astronomical-data-explosion/. [Accessed 5 March 2013].

[2] J. Marlow, "What to Do With 1,000,000,000,000,000,000 Bytes of Astronomical Data per Day," Wired, 2 Apr 2012. [Online]. Available: http://www.wired.com/wiredscience/2012/04/what-to-do-with-1000000000000000000-bytes-of-astronomical-data-per-day/. [Accessed 5 March 2013].

[3] Kanellos, "Moore's Law to roll on for another decade," CNET, 10 February 2003. [Online]. Available: http://news.cnet.com/2100-1001-984051.html. [Accessed 20 March 2013].

[4] C. Cui, F. Dongwei, Z. Yongheng, K. Ajit, H. Boliang, C. Zihuang, L. Jian and N. Deoyani, "Enhanced management of personal astronomical data with FITSManager," *New Astronomy,* vol. 17, no. 2, p. February, 2012.

[5] S. Seager, R. Dotson and Lunar and Planetary Institute, Exoplanets, Houston: University of Arizona Press, 2010.

[6] J. Pepper, R. W. Pogge, D. L. DePoy, J. L. Marshall, K. Z. Stanek, A. M. Stutz, S. Poindexter, R. Siverd, T. P. O'Brien, M. Trueblood and P. Trueblood, "KELT-North: Telescope," *Publications of the Astronomy Society of the Pacific,* vol. 119, no. 858, pp. 923-925, 2007.

[7] J. Pepper, R. B. Kuhn, R. Siverd, D. James and K. Stassun, "The KELT-South Telescope," *Publications of the Astronomical Society of the Pacific,* vol. 124, no. 913, pp. 230-241, 2012.

[8] "SurveyMonkey," [Online]. Available: http://www.surveymonkey.com/. [Accessed 13 March 2013].

[9] "Google Drive Apps," Google, [Online]. Available: https://www.google.com/intl/en_US/drive/start/apps.html. [Accessed 13 March 2013].

[10] D. L. Pollacco, I. Skillen, A. C. Cameron, D. J. Christian, C. Hellier, J. Irwin, T. A. Lister, R. A. Street, R. G. West, D. Anderson, W. I. Clarkson, H. Deeg, B. Enoch, A. Evans, A. Fitzsimmons, C. A. Haswell, S. Hodgkin, K. Horne, S. R. Kane, F. P. Keenan, P. F. L. Maxted, A. J. Norton, J. Osborne, N. R. Parley, R. S. I. Ryans, B. Smalley, P. J. Wheatley and D. M. Wilson, "The WASP Project and the SuperWASP Cameras," *Publications of the Astronomy Society of the Pacific*, vol. 118, no. 848, pp. 1407-1418, 2006.

[11] G. Pojmanski, "The All Sky Automated Survey," *ACTA ASTRONOMICA*, vol. 47, pp. 467-481, 1997.

[12] M. Tuupola, "Lazy Load Plugin for jQuery," [Online]. Available: http://www.appelsiini.net/projects/lazyload. [Accessed 5 March 2013].

[13] J. Wright, G. Marcy, California Planet Survey Consortium and O. Fakhouri, "Exoplanet Data Explorer," [Online]. Available: http://www.exoplanets.org/. [Accessed 20 March 2013].

[14] "Gapminder," Gapminder Foundation, [Online]. Available: http://www.gapminder.org. [Accessed 20 March 2013].

[15] "web2py Web Framework," [Online]. Available: http://www.web2py.com/examples/default/what. [Accessed 5 March 2013].

[16] "Numpy," [Online]. Available: http://www.numpy.org/. [Accessed 5 March 2013].

[17] "gnuplot homepage," [Online]. Available: http://www.gnuplot.info/. [Accessed 5 March 2013].

[18] "GraphicsMagick Image Processing System," [Online]. Available: http://www.graphicsmagick.org/. [Accessed 5 March 2013].

[19] M. Wojciechowski, "imgAreaSelect," odyniec.net, [Online]. Available:

http://odyniec.net/projects/imgareaselect/. [Accessed 5 March 2013].

[20] "jQuery," [Online]. Available: http://jquery.com/. [Accessed 5 March 2013].

[21] "jQuery UI," [Online]. Available: http://jqueryui.com/. [Accessed 5 March 2013].

[22] E. Bressert and T. Robitaille, "ATpy - Astronomical Tables in Python," 2009. [Online]. Available: http://atpy.github.com/. [Accessed 5 March 2013].

[23] "PyFITS," Space Telescope Science Institute, [Online]. Available: http://www.stsci.edu/institute/software_hardware/pyfits. [Accessed 5 March 2013].

[24] "astrolib," [Online]. Available: https://trac.assembla.com/astrolib. [Accessed 5 March 2013].

[25] J. Machin, "xlrd 0.9.0," Python Package Index, [Online]. Available: https://pypi.python.org/pypi/xlrd. [Accessed 5 March 2013].

[26] I. Foster, "Parallel Algorithm Examples," Designing and Building Parallel Programs, 1995. [Online]. Available: http://www.mcs.anl.gov/~itf/dbpp/text/node10.html. [Accessed 5 March 2013].

[27] "Eureqa," Cornell Creative Machines Lab, [Online]. Available: http://creativemachines.cornell.edu/eureqa. [Accessed 5 March 2013].