Robotic Telescopes, Student Research and Education (RTSRE) Proceedings
Conference Proceedings, Hilo, Hawaii, USA, Jul 23-27, 2018
Fitzgerald, M., Salimpour, S., Eds. Vol. 2, No. 1, (2019)
ISBN 978-0-6483996-1-2 / doi : 10.32374/rtsre.2019.013 / CC BY-NC-ND license
Peer Reviewed Article. rtsre.org/ojs

Google CoLaboratory as a Platform for Python Coding with Students

Kalee Tock[1]*

Abstract
Google CoLaboratory (Google CoLab) is a powerful collaborative tool for coding in Python with students. This work presents a project to calculate the period of an eclipsing binary system that was completed by Stanford Online High School students using Google CoLaboratory. The Las Cumbres Observatory 0.4m telescopes were used to obtain images, and photometry from the Our Solar Siblings pipeline was imported into Google CoLaboratory using JSON (Javascript Object Notation) for analysis in Python. Some additional classroom applications of Google CoLaboratory are highlighted, such as converting between astronomical coordinate systems.

Keywords
Keyword1 — Keyword2 — Keyword3

[1] *Stanford Online High School, California, USA*
***Corresponding author**: kaleeg@stanford.edu

Introduction

Python modules have become increasingly popular in astronomy for data analysis. Astronomers cite Python's numerous modules, extensive user community, helpful documentation, ease of use, and most of all, the powerful plotting functionality as reasons for adopting Python in their research. The developer and user community that has grown up around Python is especially dynamic and supportive, which contrasts with and augments the astronomical community of practice Greenfield (2011). Outside of astronomy, applications in geociences are increasingly using Python because it is free, accessible, and multiplatform Lin (2012).

It can be difficult to introduce students to Python because of the complications of software installation and platform differences, particularly if students are using their own individual computers. Using a browser-based tool lessens several of these difficulties. Although Google CoLaboratory does necessitate use of the Chrome browser, the lack of additional software that is needed makes it particularly useful in a classroom environment to bypass platform and software issues and get students straight into coding.

In this project, two separate student groups developed Python code to calculate the period of an eclipsing binary system (Altunin et al. 2020; Badami et al. 2020). The Las Cumbres Observatory (LCO) 0.4m telescopes were used for image acquisition, and the Our Solar Siblings pipeline was used for photometry (Brown et al. 2013; Fitzgerald 2018). We had the most success imaging eclipsing binaries with apparent magnitude $m < 13$, and selecting systems that had deep primary and secondary eclipses so as to be sure of seeing a definite dip in the lightcurve from our data. The Kepler Eclipsing Binary Catalogue hosted at Villanova University provided the list of systems from which the student groups selected their targets Kirk et al. (2016).

The Importance of Manually Inspecting Time Series Images

One important takeaway from this project was the importance of flipping through all of the images manually to determine which were and were not suitable for analysis before feeding them into the code. Images that showed evidence of atmospheric interference or of collimation problems, as shown in Figure 1, compromised the photometry. The students constructed a scale from 1 – 4 to rate the quality of their images, where 1 represented a high quality image with round, well-defined stars and 4 was an image that was corrupted by clouds, satellite trails, or imperfect tracking. They used this scale to see how the results changed when images of different quality ratings were included or excluded from the analysis.

Determining Appropriate Exposure Times for Variable Stars

Determining an appropriate exposure time is an iterative, trial-and-error sort of process. AstroImageJ (AIJ) Collins et al. (2017), a free software for manipulating astronomical images, allows the user to interactively determine the Right Ascension (RA) and Declination (DEC) of stars by simply moving their mouse over the star in the image. This is achieved by using the World Coordinate System plate solution. which allows the user to efficiently locate particular stars in the image. The user decides how big to make the circle, or aperture, and places the aperture over the star. Within the aperture, AIJ computes the centroid of the star as the average position of the pixels, weighed by a measurement of the light collected by each pixel, and automatically adjusts the aperture to be centered at the centroid.

The measurement of light is reported as a count of analog-digital-units (ADU's). When photons strike the camera CCD, electrons are knocked loose from their corresponding "buckets" on each pixel. The ADU count is the number of these electrons. AIJ reports the highest ADU count for a single pixel within the aperture as the "peak", and the total ADU count from all of the pixels within the aperture as "Int counts", or "integrated counts", shown in Figure 2. Where the centered aperture covers a fraction of a pixel, the corresponding fraction of the ADU count measured by that pixel is included in the integrated counts.

To avoid saturation, the "peak" should be comfortably lower than the total number of electrons that the bucket can hold, which is usually 65,535 but can vary depending on the telescope Buchheim (2015). However, the integrated counts needs to be high enough that the signal will not be overwhelmed by the inherent noise in the measurement. Noise comes from electrons jiggling out of their buckets due to causes that are not target star photons, like light from other sources or heat from the telescope that causes them to jiggle out without a photon stimulus. AIJ calculates the signal-to-noise (SNR) ratio from the aperture tool settings. It first counts up the photons inside the aperture radius: this is the signal. Then it subtracts off the average photons per area from the region between an inner annulus and an outer annulus that are both located outside the aperture. This is the noise. In general, an SNR between 100 and 200 is desirable, which usually corresponds to integrated counts of at least 100,000 ADU and less than 500,000 ADU Fitzgerald et al. (2018).

Another factor that must be kept in mind is that these stars are variable, which is the whole point of measuring their light in the first place. The brightening and dimming of the target star, combined with the clarity of the sky on the night the image is taken, might correspond to as much as a doubling or halving of the counts from any single image. So, it is important not only for the ADU counts to be in range, but for double or half that number of ADU counts to be in range also.

This is further complicated by the need for the ADU counts to be in range for several comparison stars (comp stars) in addition to the target star.

Comp Stars

Even though the specifications of the LCO telescopes and cameras are identical, the images are

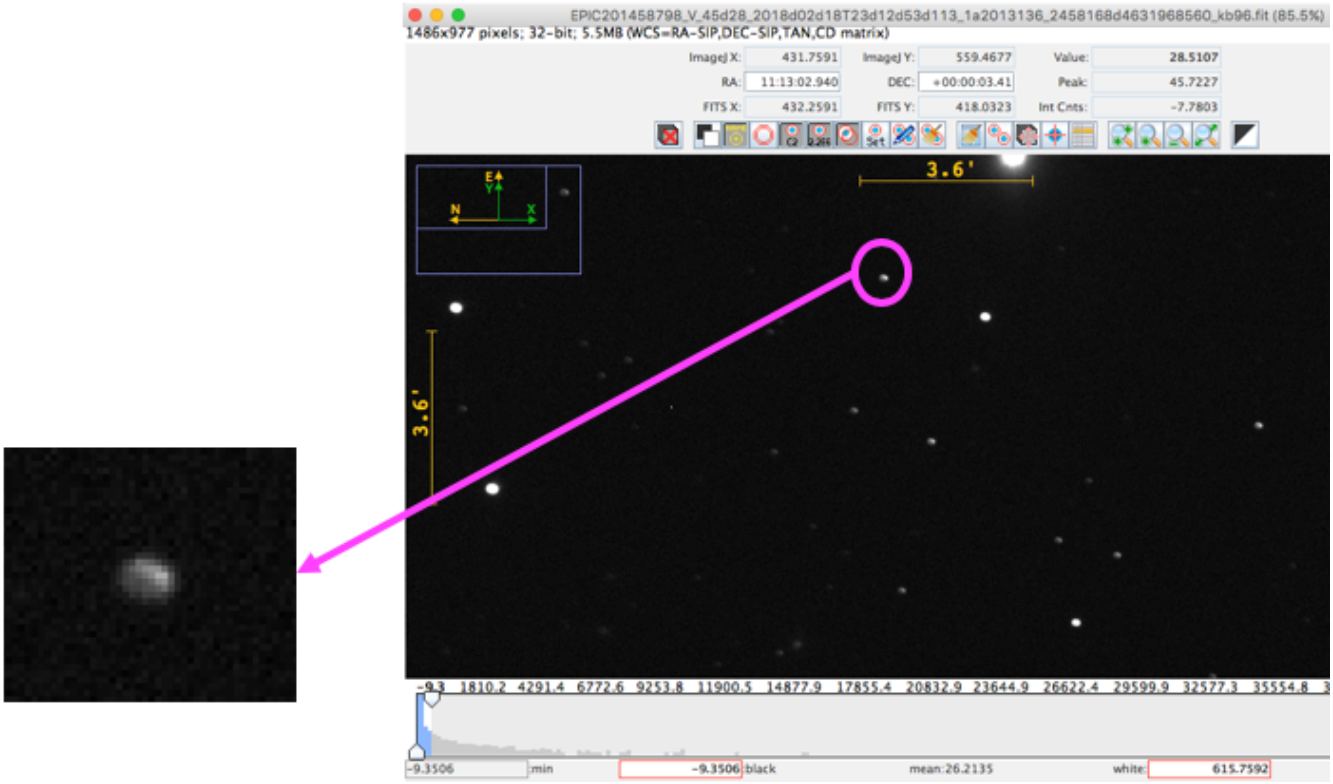

Figure 1. Sample image showing evidence of poor collimation

taken in multiple locations over the course of multiple nights Brown et al. (2013). Some skies are clearer than others, and some viewing angles are more direct than others. But, if atmospheric effects cause more or less light from the target to reach the telescope, then the atmosphere is likely to affect a nearby comp star in the same way. So, instead of reporting the measurement of light from the target directly, we report the ratio of target light to comp light. This ratio is equivalent to the difference in the star magnitudes, since magnitude is on a log scale.

Of course, this only works if the comp star is not itself inherently variable. To find the least variable comp star, the students plotted the differential magnitudes of several candidate comp stars relative to each other, looking for the flattest lines. It was helpful to arrange the plots as a matrix in order to disentangle the effect of one comp star from that another. For example, the slight swishiness of the sample comp star in Figure 3 below should not eliminate the comp star against which it was plotted, because this swishiness showed up in the plots of this comp star against all of the other candidates. In addition to examining the Comp vs. Comp plots visually, the students also compared the standard deviations of the magnitudes and the slopes from linear fits of each graph, took into account the roundness, color, and average counts from the comp star candidate compared to the target across all of the images, and examined the scatter in the final lightcurve.

Performing photometry manually in AstroImageJ on target and comps works well, and it is important to do it manually for several images to find appropriate exposure times, select comp stars, get a feel for the starfield, and understand what the numbers mean. However, each project had almost 600 images by the time it finished. It would be impractical and error-prone to perform the photometry in this way for that many images. Also, as it turns out, there is more than one way to count photons.

Photometry from the OSS Pipeline

The Our Solar Siblings (OSS) pipeline performs six types of photometry on images that are returned

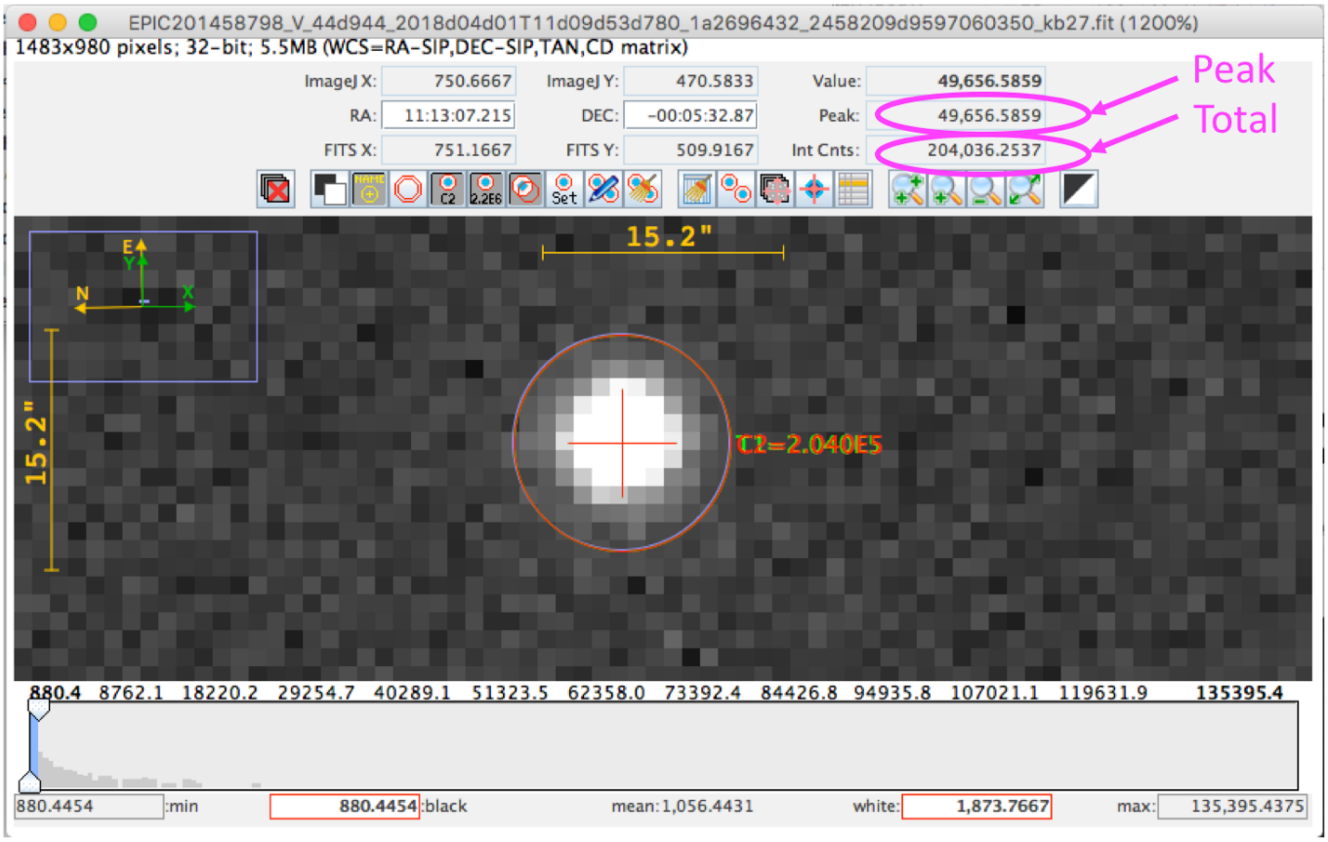

Figure 2. Aperture photometry in AstroImageJ, showing the peak ADU count and integrated counts within the aperture set by the user.

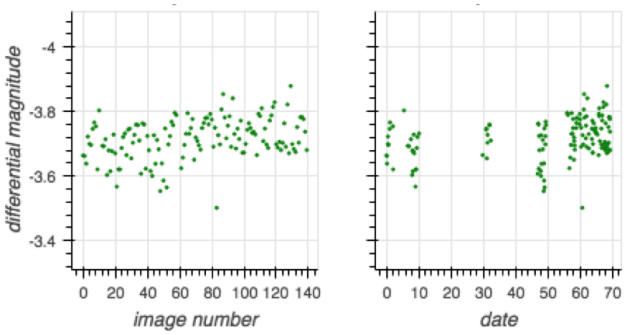

Figure 3. Sample comp star candidate eliminated on the basis of its non-linear differential magnitude plotted versus other comp stars

from LCO Fitzgerald (2018). The first three photometry types are similar to each other in that they represent straight sums of ADU counts within an aperture. These include aperture photometry (apt), which is the type of photometry described above that the students performed manually in AIJ. In addition, for each image, the pipeline returns source extractor photometry (sex), and source extractor kron photometry (sek). The sex and sek photometry algorithms measure integrated counts similar to apt, though sex varies the aperture radius for each star so as to capture 90% of the object's light, and sek models the star's image as elliptical rather than circular Holwerda (2005).

The remaining three photometry algorithms are called dao, dop, and psx. These use mathematical models called point-spread functions (PSF's) to measure the light from any given star in an image (Stetson 1987; Schechter et al. 1993; Bertin 2011; Bertin and Arnouts 1996). These algorithms operate on the premise that if the telescope were outside of Earth's atmosphere, the starlight from a target would fall on a single pixel of the CCD. Figure 4 shows the intensities of two stars A and B, where B is brighter. The horizontal axis is the pixel x-coordinate, and the intensity is shown as an ADU count spike at a single pixel.

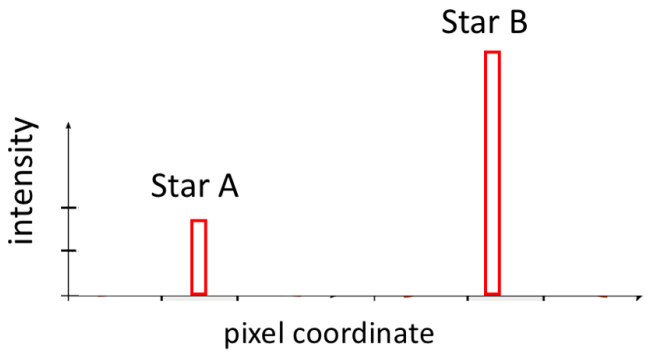

Figure 4. Starlight from two stars before passing through Earth's atmosphere, as they appear on a one-dimensional cross section of the CCD

As it passes through the atmosphere, however, the light is spread out by atmospheric blurring, optical quality and focus accuracy. The image of the star becomes a "point spread function" that is bright at the center and dimmer around the edges. Conservation of energy dictates that the peak intensity and also the total area under the point spread function are both proportional to the total number of photons from the star. Also, the full-width-at-half-maximum, or FWHM, is the same for all stars, as shown in Figure 5 Buchheim (2015).

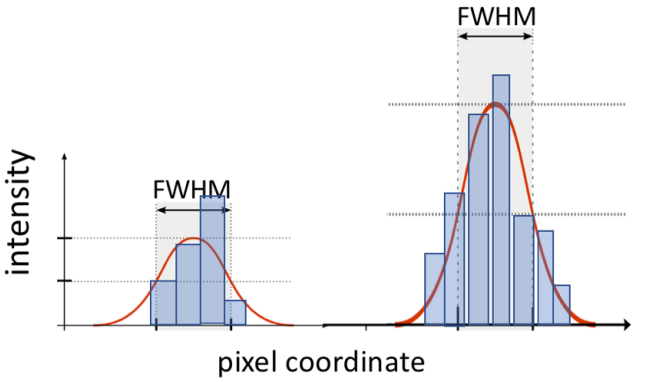

Figure 5. Starlight from two stars after passing through Earth's atmosphere, as they appear on a one-dimensional cross section of the CCD, fit to a point-spread function

All of this is usually true, before detection. However, we are collecting light in discrete buckets, pixel by pixel. Discrete buckets do not make a smooth curve, particularly if the centroid of the star is located in an awkward position relative to the pixel boundaries. So, the types of dao, dop, and psx are different mathematical models for what the shape of the smooth curve would be if the data could be taken continuously rather than discretely.

The details of dao, dop and psx, and the mathemtical methods which underpin them have been discussed in other papers (Stetson 1987; Schechter et al. 1993; Bertin 2011). We took the pragmatic approach of deciding to use the model that gave us the cleanest looking lightcurves. But to do that, it was necessary to determine how to plot the lightcurves in the first place.

Format of Data Returned By the OSS Pipeline

What it means to say that "the OSS pipeline performs 6 types of photometry" is that for every image, 6 separate, comma-delimited text files are produced. The name of each textfile contains information about the image and photometry used: target, filter, exposure time, date, airmass, telescope, and photometry type. In each of the text files are several lines, each line corresponding to one star that the photometry found in the image. The line begins with the RA and Dec of the star, and then has the integrated ADU count and the error for that star, as well as the x and y pixel coordinates of the star in the image. A schematic of the information associated with each image is shown in Figure 6.

Our goal was to locate the target star and the comp stars from these lists of stars in each image, and plot a lightcurve. But first, it was necessary to find a reliable way of storing and sharing the data with all of the students who were working on it. Also, there was more than one eclipsing binary project going: two class teams and a pilot project. So, there were tens of thousands of data lines, each pertaining to a star in a particular image.

Storing and Porting Data for Python

In order to shield students file IO, extraction of the image data from the filename using regular expressions, and designing data structures, it was helpful to design a data structure for each system that con-

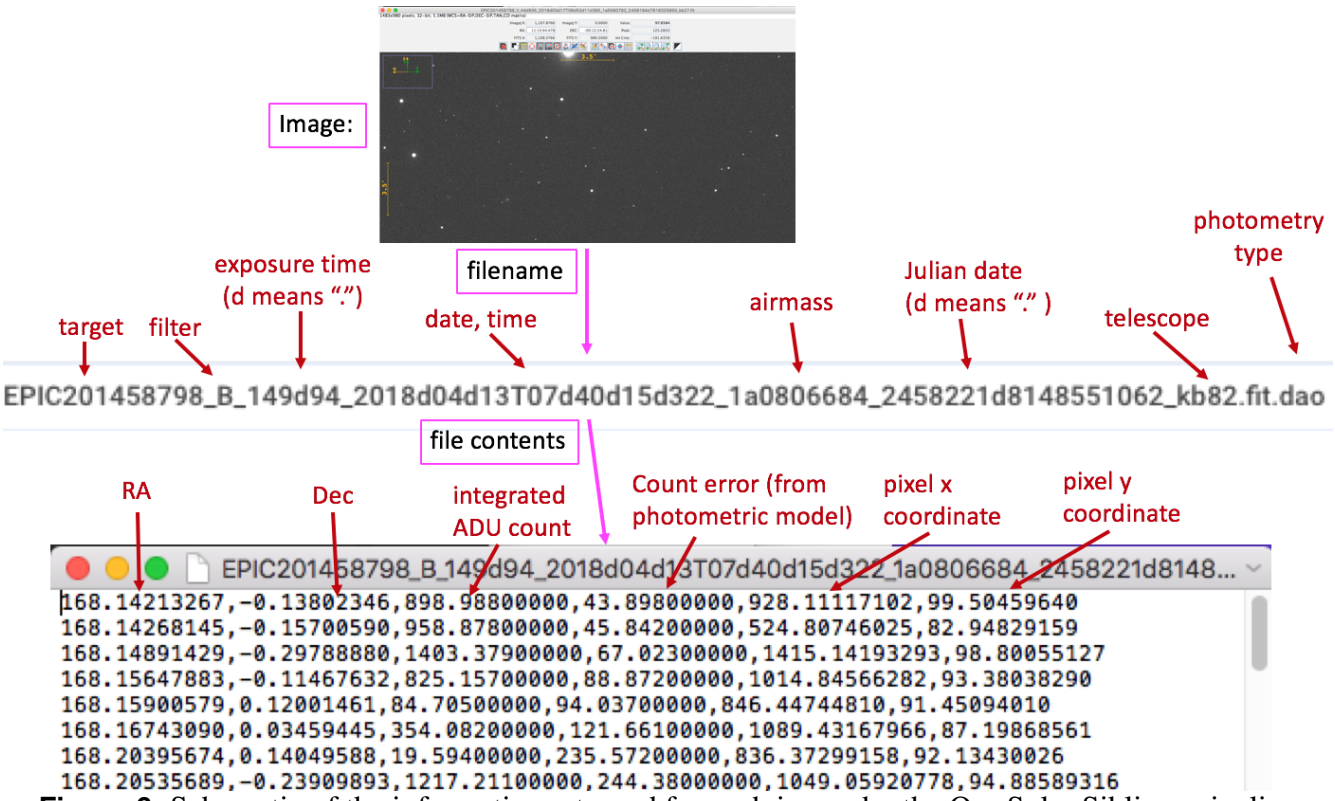

Figure 6. Schematic of the information returned for each image by the Our Solar Siblings pipeline

tained all of the information from all of the photometry files.

It is designed as a Python dictionary called "system" whose fields describe a given target. Among those fields are dictionaries for each photometry type. Each photometry type dictionary contains an array of images. Each image is itself a dictionary with fields for the filter, exposure time, date, and an array of stars. Within the star array, each star is a dictionary with fields for the RA, Dec, count, count error, and pixel coordinates. Figure 7 shows a schematic of the data structure into which the data from the files in Figure 6 get loaded.

Some simple code extracted information from the filenames and read the text files into this data structure on the instructor's home computer. Then, JSON was used to create one (enormous!) text file containing all of the data. This is accomplished with a single line of code:

```
json_file.write(json.dumps(system))
```

The text file generated by this command then gets copied to where it is needed (in my case, Google drive) and read back into the same Python data

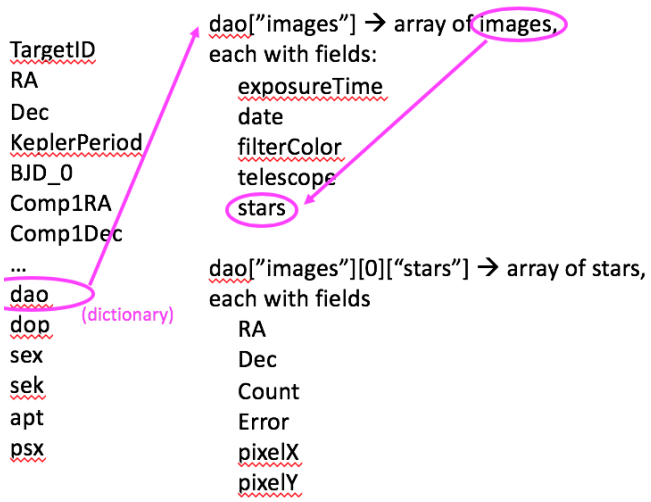

Figure 7. Schematic of the Python data structure that the students worked with

structure using the command:

```
system = json.loads(string)
```

I think this is a particularly good system for teachers, because it allows the teacher to design the data structure and make sure that all her students are using the same organization. Also, it does

shield the students from file I/O and regular expressions and things that are more computer-sciencey than astronomy-specific. Although, it could be argued that computer science and astronomy are intertwined, and students should be given more authentic experiences in their use, we must look at the practically and teacher training. The aim is to allow teachers of varying ability to take this lesson and construct a program that they are confident in delivering.

The text file generated by the json.dumps command could be examined if that were desirable, though it places enormous load on the computer memory. However, it is human-readable, and shows all of the structures and data. The beginning of the file that was generated for one of the systems is shown in Figure 8.

Google CoLaboratory

We used Google CoLaboratory for developing code to analyze our systems. CoLaboratory has the functionality of Google Docs for Python code, making it ideal for classroom use. The students were already using Google Docs to write their papers collaboratively, so having a consistent interface for writing code was helpful. To see CoLaboratory in action, open Google Drive on a Chrome browser. First, make a new CoLaboratory, by going to "File > New > More". New CoLaboratory users may need to click the "Connect More Apps" button before the yellow infinity-sign CoLaboratory option will appear.

CoLaboratory has two types of cells: code cells and text cells. Users start in a code cell by default. For example, one can type

```
print (Hello Astronomy!)
```

and then type Shift-Enter to run. The output of a cell appears below that cell in the CoLaboratory.

Text cells can be accessed by clicking the menu item at the top. Text cells allow the creation of human-readable explanations, including links and images, with an intuitive word-processing interface. As with the code cells, clicking shift-enter from within a cell displays the output text on the screen. Text cells are particularly useful for teaching because they allow integration of code with rich text describing the code and the theory behind it.

One useful application for the astronomy classroom is converting between degree and hh:mm:ss coordinates. To do this, it is necessary to install astropy by typing the following commands into a code cell:

```
!pip install astropy
from astropy.coordinates import SkyCoord
from astropy import units as u
```

In cases where it is desirable to convert a whole spreadsheet of coordinates from one format into another, or import a large JSON file into the code as was necessary for the eclipsing binary projects, we must understand how to make the CoLaboratory read data files from Google Drive.

Reading Data Files from Google Drive

The first important point about reading datafiles is that one should not do it from a personal account. The reason for this is that the code must be given permissions to do anything it wants to the account from which it reads datafiles. This is unwise if multiple people are editing code, no matter how trusted they are. It is best to make a throwaway Google drive account to store the data, for which the password can be freely shared. Note that the code does not need to be run from the throwaway Google drive account. When any CoLaboratory is run that reads files from Google drive, the user is prompted to specify an account and authenticate if necessary.

Timing Out

When a CoLaboratory is run, Google assigns a virtual machine to the user who is running it. After a long period of inactivity, it logs the user out to free up the virtual machine for other users. In practical terms, this means that after a long (where "long" generally means "a few hours") period of inactivity, the initial code cells will need to be re-run, any software like astropy will need to be re-installed, and data files will need to be re-loaded.

```
{"TargetID": "201826968", "RA_in_degrees": "178.3651", "Dec_in_degrees": "5.8594",
"KeplerPeriod": "0.3617589", "BJD_0": "1974.234886", "Comp1RA_in_HHMMSS":
"11h53m39.228s", "Comp1Dec_in_HHMMSS": "05d54m5.59s", "Comp2RA_in_HHMMSS":
"11h53m40.44s", "Comp2Dec_in_HHMMSS": "05d49m43.56s", "Comp3RA_in_HHMMSS":
"11h53m40.09s", "Comp3Dec_in_HHMMSS": "05d43m33.24s", "Comp4RA_in_HHMMSS":
"11h53m40.83s", "Comp4Dec_in_HHMMSS": "05d45m31.25s", "Comp5RA_in_HHMMSS":
"11h53m12.65s", "Comp5Dec_in_HHMMSS": "05d45m47.40s", "Comp6RA_in_HHMMSS":
"11h53m11.52s", "Comp6Dec_in_HHMMSS": "05d47m34.36s", "CompRasInDeg":
[178.41344999999998, 178.41849999999997, 178.41704166666665, 178.42012499999998,
178.30270833333333, 178.29799999999997], "CompDecsInDeg": [5.901552777777778,
5.828766666666667, 5.7259, 5.758680555555555, 5.763166666666667, 5.792877777777778],
"psx": {"images": [{"filterColor": "B", "exposureTime": "49.938",
"modifiedJulianDate": "2458170.9855853538", "telescope": "kb82", "category": "",
"comments": "", "score": "", "stars": [{"RA": "178.2362010", "Dec": "+5.7156659",
"pixelX": "1193.8076", "pixelY": "82.8025", "Count": "5184.38", "Error": "109.6183"},
{"RA": "178.5173112", "Dec": "+6.0868401", "pixelX": "18.9791", "pixelY": "960.4824",
"Count": "0", "Error": "0"}, {"RA": "178.5137592", "Dec": "+6.0683268", "pixelX":
"77.4402", "pixelY": "949.5721", "Count": "7987.723", "Error": "122.6227"}, {"RA":
"178.5119282", "Dec": "+5.7143109", "pixelX": "1194.4917", "pixelY": "948.1808",
"Count": "6433.571", "Error": "112.1856"}, {"RA": "178.5108078", "Dec": "+5.6938513",
"pixelX": "1259.0665", "pixelY": "944.8024", "Count": "3168.842", "Error":
"95.67119"}, {"RA": "178.5099907", "Dec": "+5.8028817", "pixelX": "914.9103",
"pixelY": "940.9825", "Count": "4359.611", "Error": "103.8775"}, {"RA":
"178.5097351", "Dec": "+5.7722586", "pixelX": "1011.7166", "pixelY": "940.5045",
"Count": "2665.312", "Error": "94.08871"}, {"RA": "178.5032350", "Dec": "+5.6805435",
"pixelX": "1301.1492", "pixelY": "921.2495", "Count": "5360.583", "Error":
"106.679"}, {"RA": "178.5009520", "Dec": "+5.8847504", "pixelX": "656.7752",
"pixelY": "911.6479", "Count": "5371.967", "Error": "109.0378"}, {"RA":
```


Figure 8. Initial lines of the json file generated for one of the systems described in this paper

Generating a Lightcurve from OSS Photometry Data

The tiny.cc/rtsre file contains code to load the data into Python, and construct a lightcurve in sek photometry. In order to accomplish this, it is necessary to find the closest star to the (preselected) target and (preselected) comp star in each of the images, making sure that the closest star is within at least 2 arcseconds of those star coordinates.

Bokeh

Bokeh is a powerful graphing software that outputs directly to the browser from CoLaboratory. We found it to be more useful and more accurate than matplotlib for the purpose of this project. A range of great features includes the ability to zoom in on particular regions of any graph after it is generated, as well as the ability to customize the colors and download the graph using the disc icon at the right. These features are highlighted in Figure 9 below.

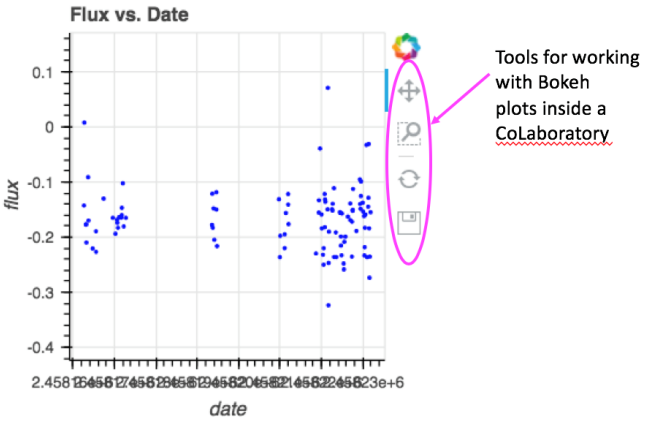

Figure 9. Sample Bokeh plot output within a CoLaboratory

Converting from Date to Phase

The reason that graph shown in Figure 9 above does not show a definite set of eclipses is that the system was sampled less than once every cycle. For the lightcurve to become apparent, the observations must be "phased", or plotted over the course of a single period. For example, if the period is 0.3 days, and the second observation occurred 0.45

days after the first observation, the second observation should be plotted at phase 0.5, because it is halfway through the period of the system relative to the first observation.

So, the way in which the lightcurve is plotted depends on the period of the system. This is problematic because the period of the system is what we are trying to calculate in the first place. As a first estimate, we use the period that the Kepler space telescope calculated for the system when it was observed 3 years ago Kirk et al. (2016). This is shown for sek photometry in Figure 10.

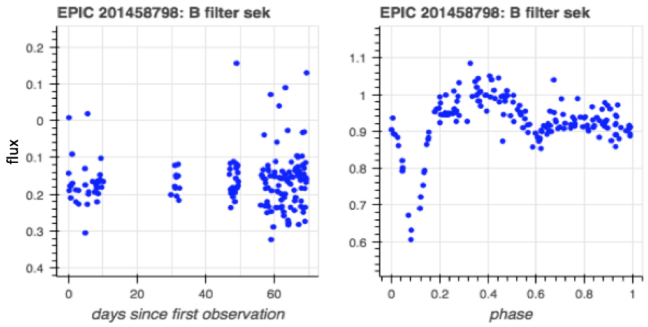

Figure 10. Example of a student eclipsing binary system plotted as flux versus days (left) and flux versus phase (right). Phase computed using period from Kepler

Selecting a Photometric Method

But, the period is not the only unknown. As explained above, the photometry is being done by the OSS pipeline in 6 different ways: dao, dop, psx, sek, sex, and apt. The reader will recall that we are taking the pragmatic approach of selecting the photometry for which the resulting lightcurve has the cleanest appearance. For one of our student groups, that turned out to be source extractor kron, or sek. For another group the standard source extractor (sex) photometry produced the best results. Some sample curves are shown in Figure 11 below.

Finding the Best Period

Having selected our photometry, the students' next task was to figure out a way to adjust the period. If the data are plotted with an incorrect period, the lightcurve looks very messy. This is evident using the Desmos tool written by Hagan Hensley. As the period is adjusted, the lightcurve changes, so that it is visually apparent when the appropriate period has been found.

Phase Dispersion Minimization: Standard Deviation and The Distance Method

However, a more mathematical justification is needed than "the lightcurve looks good". The students used two methods for finding the period mathematically.

Both of these methods operate on the premise that if the period is correct, then points on a flux-versus-phase graph that are close together in phase, will also be close in flux. Although there are parts of the curve where the flux changes rapidly with phase, points close in phase will still have fluxes that are more similar than they would be if the plots were constructed on the basis of an inappropriate period (Dworetsky 1983; Stellingwerf 1978).

The standard deviation method bins the observations into 10 groups by phase and sums the standard deviations of each bin's fluxes. The distance method sums the physical distances between adjacent-phase points on the flux-versus-phase graph. The methods are shown graphically in Figure 12. Iterating the period over multiple possible values and minimizing the respective sums gives the best period.

Fortunately, both methods yielded a clear winner for period. An example is shown in Figure 13. Also, it turned out that the minimum gave a period that was identical to the period provided via Kepler's dataset to within a few seconds. Plotting the lightcurve with this minimum period indeed yielded a very well-defined lightcurve. This gave us confidence in our method of calculating it.

Computing the Error

The literature is conflicting regarding computation of the error of a period obtained by phase dispersion minimization as we have done (Montgomery and Odonoghue 1999). Our best solution was to take as our error the distance in period-space between

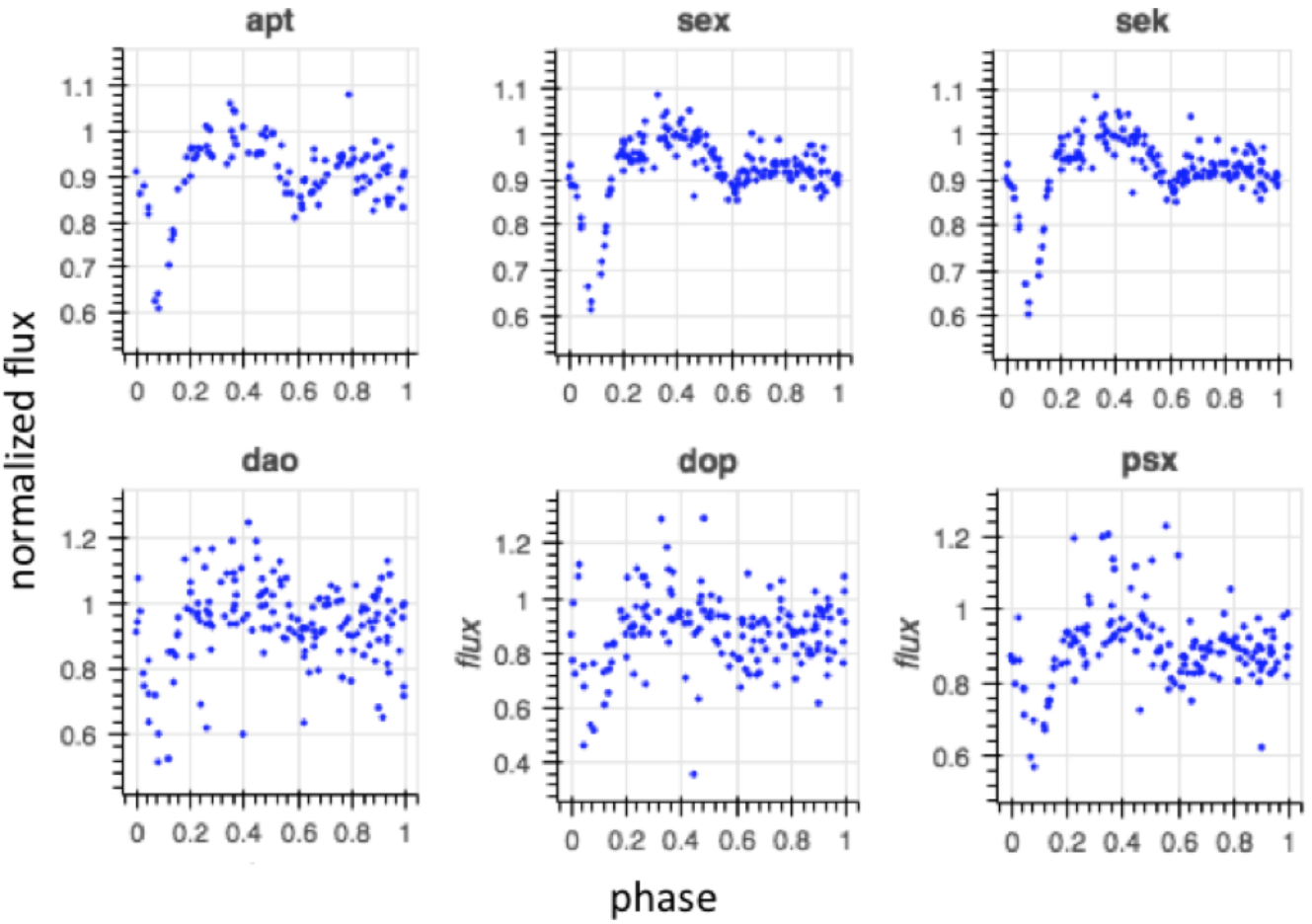

Figure 11. Sample lightcurves from the various photometry types, used to process 2x2 binned images from the LCO 0.4m telescopes. Phase computed using period from Kepler

the two points that fall 5% of the way up from the minimum point of the curve shown in Figure 13. Although 5% seems somewhat arbitrary, this does give the reader an idea of the range of periods that would yield a relatively low standard deviation or distance.

Future Work: Time Series Projects

Time series photometric analysis such as that done in these projects is useful for investigating eclipsing binaries, and that is what was done in these projects. However, there are some other types of variable star projects that could make use of the code that was developed here. The students are inspired by and excited about the possibility of studying exoplanets, which also cause a predictable variation in starlight when they pass in front of their host star. Other stars whose characteristics can be understood through photometry include RR Lyrae, which grow and shrink in a predictable cycle due to changing surface temperature.

The students would also like to re-examine the method used for determining the period error in this work, as they entertained some (well-founded) misgivings regarding the 5% method that we used. Finally, they would like to understand the temperature of the system using B – V curves (Sekiguchi and Fukugita 2000) . The instrumental B – V curve for the system featured here, shown in Figure 14, seems too noisy to be productive for further analysis. However, we have been pointed toward some modeling software that might be able to make sense of it.

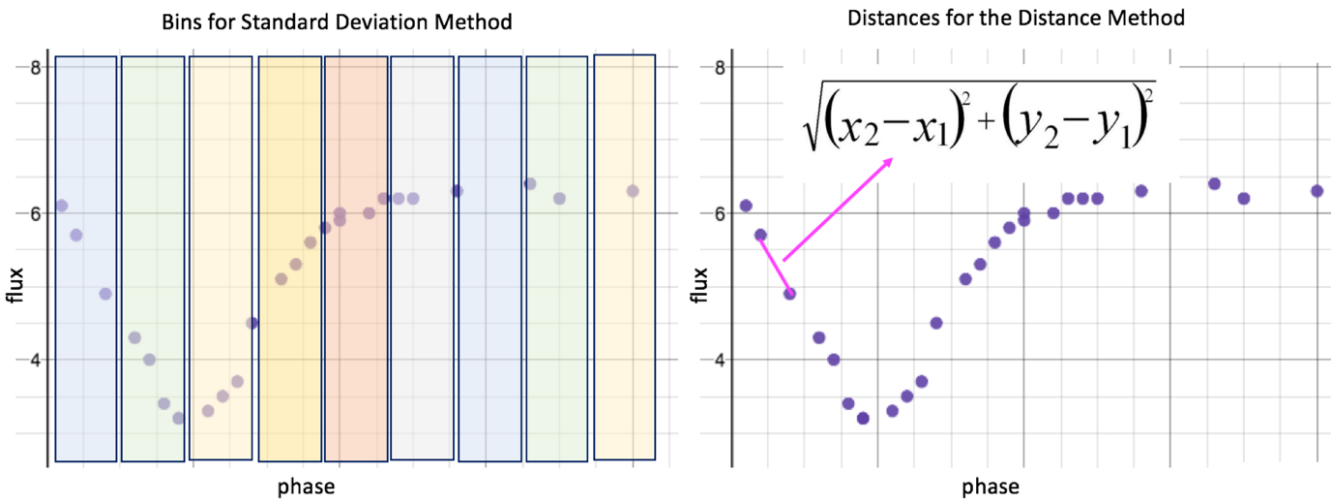

Figure 12. An illustration of the quantities to be minimized for the standard deviation and distance methods, respectively.

Future Work: CoLaboratory

One of the participants at the 2018 RTSRE conference suggested that there is an easier way to share datafiles online, using something called Firebase cloud storage. Apparently, if we use firebase, we will be able to access the file from Python directly using a "requests.get" command, without having to give anyone permissions to any Gdrive account. So, possibly the contortions around filesharing will soon become more straightforward.

Also, I would like to find a way to flip through the raw .fits images within a CoLaboratory, so as to be able to rate their quality directly in the browser by scrolling without having to keep track of this separately in a spreadsheet. This does not appear to be possible currently, but I am looking for a solution.

Conclusion

Using Python code written together in Google CoLaboratory, students were able to investigate the photometry of images of an eclipsing binary system that were returned by the Our Solar Siblings pipeline. Google CoLaboratory is a powerful system for enabling students to perform such investigations, because it facilitates students writing code together at the same time in a shared document, viewing and learning from each others' techniques and each others' error messages.

Acknowledgments

I would like to thank my husband Theron for introducing me to JSON and for generally being my IT (and personal) support. My sons Ezra and Ryan helped me with Python syntax on multiple occasions (no, I am not too proud to ask for help from an 11-year old. Desperate times.). Special thanks go to Rachel Freed for introducing me to the wonderful world of astronomy research, for being an amazing mentor herself and for connecting me to other fantastic mentors. One of those was Michael Fitgerald, of Our Solar Siblings, a frequent source of invaluable help throughout this project and others. Thanks to Richard Harshaw and Bob Buchheim for patiently explaining how image saturation works and to Russ Genet for his encouragement. Thanks to the Las Cumbres Observatory robotic telescope network, which generously provided the telescope time and much-appreciated support throughout this project. Thanks especially to my incredible science division colleagues and the amazing students of Stanford Online High School, to whom I am grateful for plunging into this journey and sticking with it (and with me) throughout its many twists and turns!

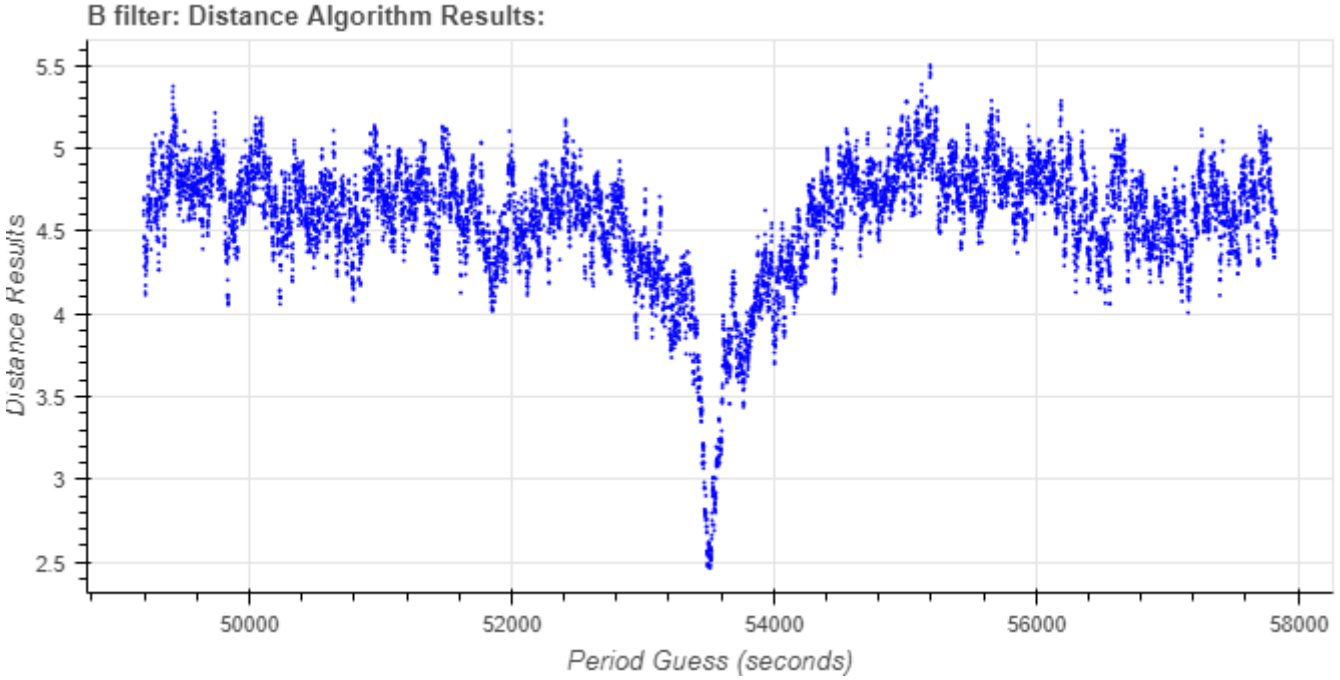

Figure 13. Sample results of the distance algorithm from one of the student projects

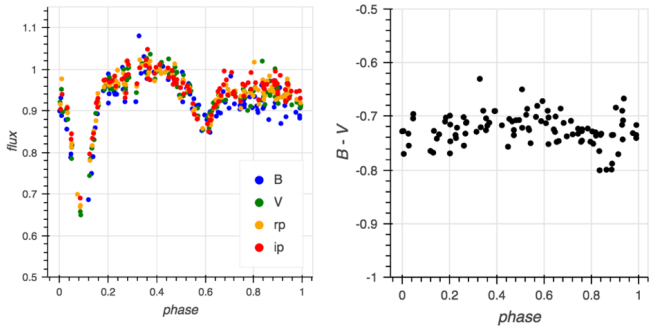

Figure 14. Sek photometry lightcurves for B, V, rp, and ip filters using Kepler Period (left) and instrumental B – V curve using sek photometry and Kepler Period (right)

References

Altunin, I., Caputo, R., and Tock, K. (2020). Period of Eclipsing Binary EPIC 201458798. *Astronomy Theory, Observations & Methods*, 1(1).

Badami, U. A., Gosart, L., North, J., and Tock, K. (2020). Observations of Eclipsing Binary EPIC 201826968. *Astronomy Theory, Observations & Methods*, 1(1).

Bertin, E. (2011). Automated morphometry with SExtractor and PSFEx. In *Astronomical Data Analysis Software and Systems XX*, volume 442, page 435.

Bertin, E. and Arnouts, S. (1996). SExtractor: Software for source extraction. *Astronomy and Astrophysics Supplement Series*, 117(2):393–404.

Brown, T., Baliber, N., Bianco, F., Bowman, M., Burleson, B., Conway, P., Crellin, M., Depagne, É., De Vera, J., Dilday, B., et al. (2013). Las Cumbres Observatory global telescope network. *Publications of the Astronomical Society of the Pacific*, 125(931):1031.

Buchheim, R. K. (2015). *Astronomical Discoveries You Can Make, Too!: Replicating the Work of the Great Observers*. Springer.

Collins, K. A., Kielkopf, J. F., Stassun, K. G., and Hessman, F. V. (2017). AstroImageJ: Image processing and photometric extraction for ultra-precise astronomical light curves. *The Astronomical Journal*, 153(2):77.

Dworetsky, M. (1983). A period-finding method for sparse randomly spaced observations or "How long is a piece of string?". *Monthly Notices of the Royal Astronomical Society*, 203(4):917–924.

Fitzgerald, M. T. (2018). The Our Solar Siblings Pipeline: Tackling the data issues of the scaling problem for robotic telescope based astronomy education projects. *RTSRE*, 1(1):347–358.

Fitzgerald, M. T., McKinnon, D. H., Danaia, L., Cutts, R., Salimpour, S., and Sacchi, M. (2018). Our Solar Siblings. A high school focussed robotic telescope-based astronomy education project. *RTSRE*, 1(1):221–235.

Greenfield, P. (2011). What python can do for astronomy. In *Astronomical Data Analysis Software and Systems XX*, volume 442, page 425.

Holwerda, B. W. (2005). Source extractor for dummies v5. *arXiv preprint astro-ph/0512139*.

Kirk, B., Conroy, K., Prša, A., Abdul-Masih, M., Kochoska, A., MatijeviČ, G., Hambleton, K., Barclay, T., Bloemen, S., Boyajian, T., et al. (2016). Kepler eclipsing binary stars. VII. The catalog of eclipsing binaries found in the entire Kepler data set. *The Astronomical Journal*, 151(3):68.

Lin, J. W.-B. (2012). Why Python is the next wave in earth sciences computing. *Bulletin of the American Meteorological Society*, 93(12):1823–1824.

Montgomery, M. and Odonoghue, D. (1999). A derivation of the errors for least squares fitting to time series data. *Delta Scuti Star Newsletter*, 13:28.

Schechter, P. L., Mateo, M., and Saha, A. (1993). DOPHOT, a CCD photometry program: Description and tests. *Publications of the Astronomical Society of the Pacific*, 105(693):1342.

Sekiguchi, M. and Fukugita, M. (2000). A Study of the B- V Color-Temperature Relation. *The Astronomical Journal*, 120(2):1072.

Stellingwerf, R. F. (1978). Period determination using phase dispersion minimization. *The Astrophysical Journal*, 224:953–960.

Stetson, P. B. (1987). DAOPHOT: A computer program for crowded-field stellar photometry. *Publications of the Astronomical Society of the Pacific*, 99(613):191.