ceph_tools Instructions for Use

Overview

The ceph_tools Python package is a set of programs that I wrote in 2018-2019 for my work on cepheid variables. These programs now exist within the rotsehub rotseana Python package. I have documented here instructions for setting up the use of ceph_tools as well as descriptions for each program and instructions for executing them. Please feel free to contact me with any questions at grantdonnelly2@gmail.com

-Grant Donnelly

Set Up

As a part of the rotseana Python package, ceph_tools can be installed on your own machine, or can be run by remote access on the ROTSE ixchel computer.

Installation on your own machine

First, make sure that Python is installed on your computer. I would recommend using the Anaconda distribution (https://www.anaconda.com/products/individual) as it will contain all of the libraries necessary to run all ceph_tools programs. Otherwise, you will need to have the following libraries installed: numpy, matplotlib, scipy, and astropy.

The rotseana Python package can be downloaded from the rotsehub page on github. You can follow this link: https://github.com/rotsehub/rotseana. From there, click the green "Clone or download" button to download. Once downloaded, open a terminal and navigate to the directory in which rotseana is located on your computer. Use the cd command to change the directory and the ls command to view the contents of the current directory to do this. Now navigate to ceph_tools using

```
> cd rotseana/py/rotseana/vsp/ceph_tools/general
```

Type and enter ls and you will see the Python programs of ceph_tools. If you plan to use lccal.py, create a directory here in which to store your match structures. You can do this with the mkdir command. Copy the match structures you will be using into this new directory. To run a program, follow the instructions for that program below. Note that all programs are run by typing python3 and then the name of the program with its inputs.

Using ixchel remotely

All of the rotseana Python package is installed on the ROTSE ixchel computer, and you can use the programs through remote access to the computer. ixchel is protected by a firewall, so you will first need to remote access a proxy computer, and then access ixchel from there. The neo computer will work fine as a proxy, so to remote access, open a terminal and enter

```
> ssh smurotse@neo.physics.smu.edu
```

Enter the password. If you don't know the password, ask another VSP team member. From neo, you can remote access ixchel. Enter

```
> ssh smurotse@ixchel.physics.smu.edu
```

And enter the password. Now you can navigate to ceph_tools

```
> cd prj/rotse/software/rotsehub/rotseana/py/rotseana/vsp/ceph_tools/general
```

If you enter the ls command you should see the contents of ceph_tools along with other directories which users have made to store their match structures. If you plan to use lccal.py, you will need to either make one of these directories yourself, or know the name of an existing directory which hold the match structures you will be using. If you need to make a new one, use the mkdir command to set up a new directory, and then copy the match structures you will be using into this new directory. You are now ready to use any of the programs by following the instructions below. Note that all programs are run by typing python3 and then the name of the program with its inputs.

lccal.py

Description

"lccal" means lightcurve calibration. The purpose of this program is to calibrate the light curve of an object between different match structures by comparing the photometry of nearby reference stars. The program outputs the calibrated lightcurve of your target in the terminal, and writes the lightcurve to a file which will be stored in the directory that your match structures are stored in.

Usage

To run lccal.py, you need to have a directory containing the match structures you want to use. This directory must be located in the same directory as lccal.py.

Inputs:

match_structures: the name of the directory in which your match structures are stored.
vra: the right ascension coordinate of your target object in decimal degrees.
vdec: the declination coordinate of your target object in decimal degrees.
radius: the radius (in degrees) around your target object in which reference stars will be searched for. This input is optional, set to 0.1 degrees by default. Increase it if you have an insufficient number of reference stars.
plots: set whether to show plots of light curves. Set to True to show plots, set to False to disable. This input is optional, set to True by default. If this input is made, the user must also enter an input for radius.

```
> python3 lccal.py match_structures vra vdec radius plots
```

Examples

Assume the directory with your match structures is called matchstr, and the location of your object is RA: 27.6176 DEC: 36.5803.

Run the program with radius=0.1 degrees (default) and with plots:

```
> python3 lccal.py matchstr 27.6176 36.5803
```

Run the program with radius=0.15 degrees and no plots:

```
> python3 lccal.py matchstr 27.6176 36.5803 0.15 False
```

Run the program with radius=0.1 degrees (default) and no plots:

```
> python3 lccal.py matchstr 27.6176 36.5803 0.1 False
```

If plots are turned on, the user will periodically be shown plots that must be closed in order to continue the program.

unconex.py

Description

“unconex” means unconfirmed exclude. The purpose of this program is to take in a light curve, group data points based on their proximity in time, and then check for consistency. This program outputs a file with the filtered light curve into the directory in which unconex.py is stored.

Usage

To run unconex.py, you will need a .dat file with the lightcurve for your object saved in the same directory in which unconex.py is stored. The rows of the .dat file must be in timestamp magnitude error format. Files that are output from lccal.py are compatible with unconex.py.

Inputs:

data: the name of the lightcurve .dat file.
diff: the acceptable difference (in days) in timestamps between data points.
kind: whether the data is from ROSTE1 or ROTSE3. If ROTSE1 enter orphans, if ROTSE3 enter sn.

```
> python3 unconex.py data diff kind
```

Examples

Assume the name of your data file is lightcurve.dat, the acceptable difference is 0.002 days.

Run the program for ROTSE1 data:

```
> python3 unconex.py lightcurve.dat 0.002 orphans
```

Run the program for ROTSE3 data:

```
> python3 unconex.py lightcurve.dat 0.002 sn
```

asassn_fmt.py

Description

“asassn_fmt” means ASAS-SN format. The purpose of this program is to take lightcurve files downloaded from the ASAS-SN database and quickly format them into .dat files that are compatible with unconex.py, lcvis.py, and psigma.py. The data in files output from asassn_fmt.py can be easily combined with other lightcurves that the VSP uses.

Usage

To run asassn_fmt.py, you will need a .csv file that you have downloaded from the ASAS-SN database.

Inputs:

datafile: the .csv file to be examined.
output_name: the name of the output file.
kind: the passband to take data from. This input is optional, set to V by default.

```
> python3 asassn_fmt datafile output_name kind
```

Take a look at the .csv file to determine which passband you want to take data from. This can be found in the column headed as "filter" in the .csv file. The kind argument needs to match the entry exactly, it is case sensitive.

Examples

Assume the name of your input data file is ASASSN_lightcurve.csv and you want your output file to be named VSP_lightcurve.

Run the program for the V passband (default):

```
> python3 asassn_fmt.py ASASSN_lightcurve.csv VSP_lightcurve
```

Run the program for the g passband:

```
> python3 asassn_fmt.py ASASSN_lightcurve.csv VSP_lightcurve g
```

lcvis.py

Description

"lcvis" means lightcurve visual. This is a graphical program which is used to view a lightcurve phased with a given period.

Usage

To run lcvis.py, you will need a .dat file with the lightcurve for your object stored in the same directory in which lcvis.py is stored. The rows of the .dat file must be in timestamp magnitude error format. Files output from lccal.py and unconex.py are both compatible with lcvis.py.

Inputs:

data: the .dat file to be examined.

```
> python3 lcvis.py data
```

Upon running the program, a window should appear with some buttons, text boxes, and a slider. The plot is showing the phase plot of the input light curve with the current period in days. The current period is always shown at the top of the window. The first way to change the period is to just type in the wanted period in the appropriate box and press enter. The "add" and "subtract" buttons can be used to add or subtract from the current period an amount equal to the current "interval," which can be set by the user. The last way to change the period is by using the slider at the bottom, which also changes by an amount equal to "interval." The text boxes to the right and left of the slider set the minimum and maximum allowed values for the slider.

Example

Assume the name of your data file is lightcurve.dat.

```
> python3 lcvis.py lightcurve.dat
```

psigma.py

Description

“Psigma” means period sigma. This program is used to search for the best period of a variable within a very narrow range of possible periods and return the error on the period measurement. This program is not used to find the period if the period is unknown, it will only work for a narrow range of periods near a known approximate period. Furthermore, psigma currently is only useful for analysis of light curves which can be closely approximated as a sine/cosine wave. This program works by performing a chi squared minimization of the goodness of fit of the light curve to a sine wave. psigma.py outputs into the terminal the best period that it found along with the associated error.

Usage

To run psigma.py, you will need a .dat file with the lightcurve for your object stored in the same directory in which psigma.py is stored. The rows of the .dat file must be in timestamp magnitude error format. Files output from lccal.py and unconex.py are both compatible with psigma.py.

Inputs:

data: the .dat file to be examined.
guess: the initial guess of the period in days.
width: the range plus or minus the guess (in days) within which the program will run.
amplitude: the amplitude (in magnitude) of the light curve.
step_size: the resolution (in days) of the periods being tested. This input is optional, set to 0.000000001 days by default.
plots: set whether to show plots of light curves. Set to True to show plots, set to False to disable. This input is optional, set to True by default. If this input is made, the user must also enter an input for step_size.

```
> python3 psigma.py data guess width amplitude step_size plots
```

If plots are turned on, the user will periodically be shown plots that must be closed in order to continue the program.

Examples

Assume the name of your data file is lightcurve.dat, the guess for period is 0.345678 days, you want to examine the range of periods 0.01 days above and below your guess, and the light curve has an amplitude of 0.15 magnitudes.

Run the program with step_size=0.000000001 (default) and show plots.

```
> python3 psigma lightcurve.dat 0.345678 0.01 0.15
```

Run the program with step_size=0.00001 (default) and hide plots.

```
> python3 psigma lightcurve.dat 0.345678 0.01 0.15 0.00001 False
```

Run the program with step_size=0.000000001 (default) and hide plots.

```
> python3 psigma lightcurve.dat 0.345678 0.01 0.15 0.000000001 False
```