Ph 21 Homework 1

Emily Springer

January 16, 2020

Assignment 1.

I used http://nesssi.cacr.caltech.edu/cgi-bin/getcssconedbid_release2.cgi for the CRTS data. I chose parameters of:

Name: her x-1

Database: Photcat

Output: VOTable

Format: short

Plot: lightcurves/locations (limit 75).

I acessed the web page using python through the lines of

$$
\begin{aligned}
&url = \text{`}http://nesssi.cacr.caltech.edu/cgi-bin/getcssconedbid\_release2.cgi\text{'} \\
&values = \{\text{`}Name\text{'} : \text{`}her\ x-1\text{'}, \\
&\qquad \text{`}OUT\text{'} : \text{`}vot\text{'}, \\
&\qquad \text{`}DB\text{'} : \text{`}photcat\text{'}, \\
&\qquad \text{`}SHORT\text{'} : \text{`}short\text{'}, \\
&\qquad \text{`}PLOT\text{'} : \text{`}plot\text{'}\} \\
&data = urllib.parse.urlencode(values) \\
&data = data.encode(\text{'}ascii\text{'}) \\
&req = urllib.request.Request(url, data)
\end{aligned}
$$

I could use the *req* object to read the html page through

$$
\begin{aligned}
&with\ urllib.request.urlopen(req)\ as\ response: \\
&\quad the\_page = response.read()
\end{aligned}
$$

the_page is of type byte. To convert it to a string, I decoded it using

$$pageDecoded = the\_page.decode(\text{“}utf-8\text{”})$$

Now I could use string manipulations to get only the data table part of the string of html code. I recognized that the data table began where the first instance of "[" appeared, and it ended right before the first "}". I decided to use this in my algorithm for picking out only the data table. I looped through every character in the string, and once it was a "[", I started adding it to my dataString variable, ending when I saw "}".

```
dataString = ""
beginning = False
end = False
for char in pageDecoded:
    if char == '[':
        beginning = True
    if beginning:
        if char == '}':
            end = True
            break
        if not end:
            dataString += char
```

To convert this string into an array of individual coordinates, I did

$$dataArray = dataString.strip('[],').split(', [')$$

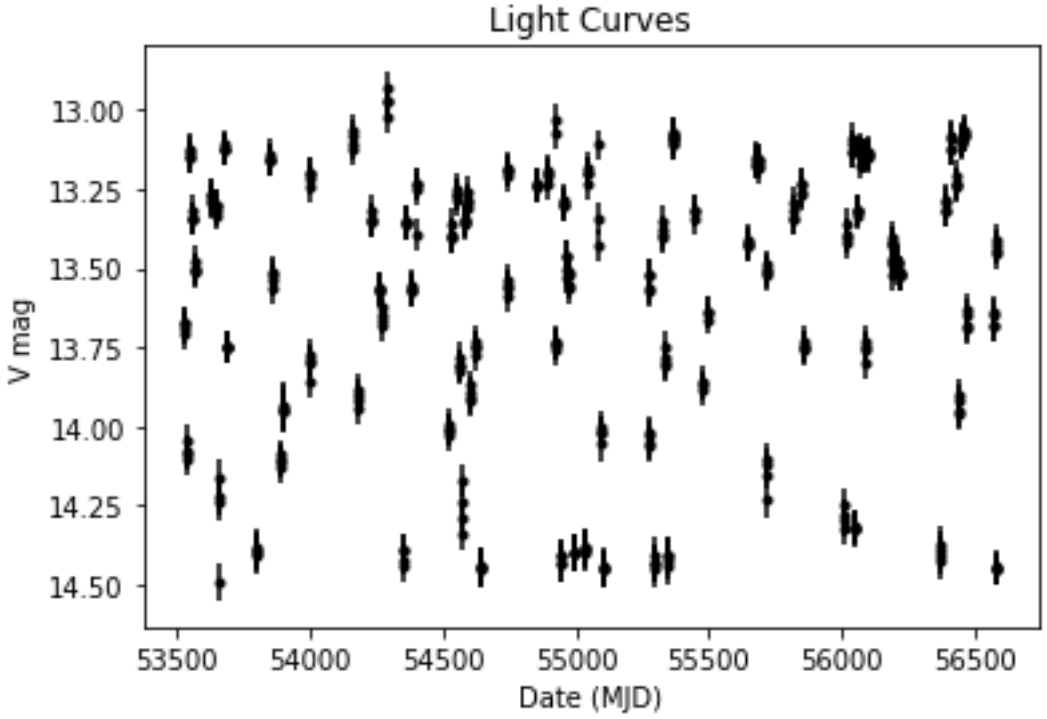
dataArray now contains an array of strings of coordinates, such as "56588.1032, 14.45, 0.05". Next, I split each number in the string into its own array to have an xArray, yArray, and yError. I did this by initializing empty arrays for each and then looping over the *dataArray*, splitting each string in the array at the ', ' to have a subarray of just the numbers. I could them easily add these numbers into the corresponding arrays, casting them to floats.

```
xData = []
yData = []
yError = []
for array in dataArray:
    point = array.split(', ')
    xData.append(float(point[0]))
    yData.append(float(point[1]))
    yError.append(float(point[2]))
```

The final step was to plot these, mimicking the look of the graph from the web by adding axes titles and error bars. I also reversed the direction of the y-axis so the numbers are decreasing since that is how it was graphed on the web.

```
plt.errorbar(xData, yData, yerr = yError, fmt = '.k')
plt.xlabel('Date(MJD)')
plt.ylabel('Vmag')
plt.title('LightCurves')
plt.gca().invert_yaxis()
```

Here is the resulting graph.



Assignment 2.

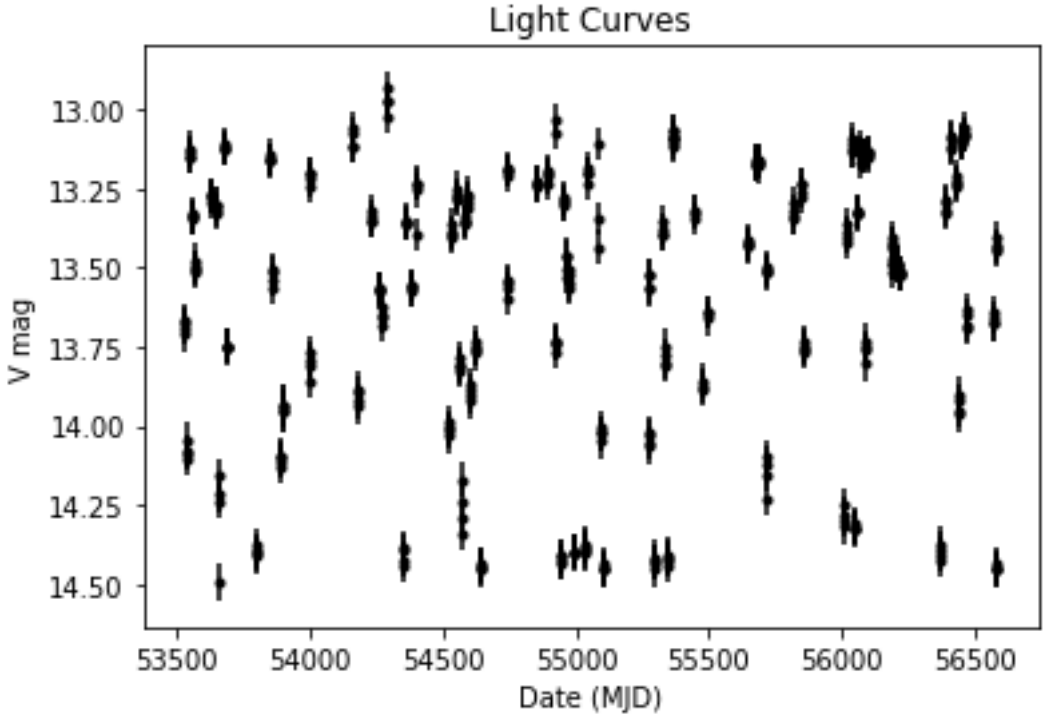
Using the same parameters as from Assignment 1, I got the VOTable file by clicking the "(right-mouse-click and save as to download)" at the bottom of the screen. This led me to http://nesssi.cacr.caltech.edu/DataRelease/upload/result_web_file47eO6v.vot, which I could use to parse through. I used a method to get the first table, which then I converted to an array.

$$votable = parse(\\ \text{“}http://nesssi.cacr.caltech.edu/DataRelease/upload/result\_web\_fileUVw9H2.vot\text{”}\\ ,pedantic = False)\\ table = votable.get\_first\_table()\\ data = table.array$$

With this format, I didn't have to add each coordinate to its own array to plot. I could instead use VOTable notation. I found what each coordinate was named from the .vot file. I could then plot it with

$$plt.errorbar(data[`ObsTime'],\ data[`Mag'],\ yerr = data[`Magerr'],\ fmt = `.k')\\ plt.xlabel(`Date(MJD)')\\ plt.ylabel(`Vmag')\\ plt.title(`LightCurves')\\ plt.gca().invert_yaxis()$$

The resulting graph was



which is the exact same graph as from Assignment 1, since the parameters were all the same.

A full copy of my entire code for both assignments can be found on my git repository at espringer0/Ph21.