Python for Variable Star Astronomy a status update

Matt Craig
Department of Physics and Astronomy

Acknowledgements

- Current students
  - Isobel Snellenberger
  - Madelyn Madsen
- Former students
  - Adam Kline
  - Erin Aadland
  - Andy Block
  - Jane Glanzer
  - Elias Holte
  - Laura Maixner
  - Stefan Nelson
  - Elizabeth Dougherty
  - Nathan Walker
  - Laura Herzog
  - Michael Meraz
  - Connor Stotts
  - Nathan Heidt
- Colleagues (MSUM)
  - Juan Cabanela
  - Linda Winkler

Outline

- Motivation
- Educational materials
- Image viewer
- Source detection and photometry
- Data reduction
- VSP

Motivation

- Prepare undergraduate students for career
  - Include some programming
  - Choose accessible language
  - Re-use as much community software as possible
- Do science: TESS/exoplanet follow-up

Goals

- Software interoperability
  - Read and write AstroImageJ photometry files
  - Generate/read AIJ source lists
- Data "interoperability"
  - TESS-style for reporting TESS data
  - AAVSO-style for reporting variable star data
- Let others do the hard work
  - Use the community-based astro software large institutions are developing
  - Make user interaction browser-based, built on tools widely used outside astro

Guide to Data Reduction

- http://bit.ly/ccd-guide

CCD Data Reduction Guide

1. Understanding astronomical images
2. Overscan and bias images
3. Dark current and dark frames
4. Flat fielding
5. Calibrating science images
6. Finding and dealing with bad pixels

Image viewer

- Browser-based; can run
  - on cloud server
  - on local machine
- Interaction with browser and Python uses
  - Jupiter framework
  - widely support by data science and finance

Image viewer

localhost:8889/lab

File Edit View Run Kernel Tabs Settings Help

Launcher | viewer-seeing-profile-Copy

Markdown | Python 3

Run all cells in this notebook

Once they run, follow the instructions near the bottom for using the viewer/setting the image that is loaded

[ ]:
```
import numpy as np
from astrowidgets import ImageWidget
from photutils import centroid_com
import ipywidgets as ipw
from astropy.stats import sigma_clipped_stats
from astropy.table import Table
%matplotlib inline
import matplotlib.pyplot as plt
from seeing_profile_functions import *
```

[ ]:
```
iw = ImageWidget(image_width=500, image_height=500, use_opencv=True)
```

[ ]:
```
np.seterr(all='ignore')
```

[ ]:
```
big_box = box(iw)
build(iw)
bind_map = iw._viewer.get_bindmap()

# Displays the event map...
#bind_map.eventmap


gvc = iw._viewer.get_canvas()
bind_map.map_event(None, ('shift',), 'ms_left', 'cursor')
gvc.add_callback('cursor-down', make_show_event(iw))
```

The cell below displays the viewer

Left-click and drag to pan

Scroll or use the +/- keys to zoom in/out

Ctrl-shift-click to display seeing profiles for stars

[ ]:
```
big_box
```

Run cell below to load/change image

[ ]:
```
iw.load_fits('ev-uma-035R.fit')
```

0 2 Python 3 | Idle Mode: Command Ln 1, Col 1 viewer-seeing-profile-Copy1.ipynb

Image viewer

- Viewer design
  - Python designed to allow multiple implementations in future
  - STScI considering its use internally
  - LSST considering plugging their viewer into this framework

Aperture photometry

- Source detection using sample image
- Perform aperture photometry on all images in folder for those sources

Photometry

MINNESOTA STATE UNIVERSITY MOORHEAD

localhost:8889/lab

File Edit View Run Kernel Tabs Settings Help

viewer-seeing-profile-Copy | photometry_directory-Copy

Code | Python 3

```
[ ]: from pathlib import Path

     import numpy as np

     from astropy.nddata import CCDData
     from astropy.coordinates import SkyCoord
     from astrowidgets import ImageWidget

     from glowing_waffles.photometry import detect_sources, faster_sigma_clip_stats, find_too_close, clipped_sky_per_pix_stats, add
```

Settings that need to be filled in by user

Fill in the names below

For the directory name you can use either a relative or absolute path.

```
[ ]: directory_with_images = '.'

     #This could be any image from roughly the middle of the sequence.
     sample_image_for_finding_stars = 'ey-uma-035R.fit'

     # This only used to center the image viewer on the object. If the object you have can't be looked
     # up in simbad this cell will raise an error. If it raises an error leave the object of interest as an empty string.
     object_of_interest = 'ey uma'

     if object_of_interest:
         obj_coords = SkyCoord.from_name(object_of_interest)

     # The name of your photometry file should include the night number (e.g. 2018-05-21), the object name
     # (e.g. kelt-16b) and the aperture radius (e.g. 12), so maybe: 2018-05-21-kelt-16b-aperture-12
     name_of_output_file = '2014-12-29-ey-uma-9 DEMO' + '.fits'
```

Set FWHM, pick aperture and annulus sizes

Using the `viewer-seeing-profile` notebook, determine what you want the FWHM should be set to. You want an aperture about 2×FWHM, an annulus with an inner radius at least 10 pixels larger than that, and outer radius about 15 pixels larger than the inner.

```
[ ]: # FWHM is used in the source detection algorithm
     FWHM = 9

     aperture_rad = 18
     inner_annulus = aperture_rad + 15
     outer_annulus = inner_annulus + 15
```

0 | 4 | Python 3 | Idle | Mode: Edit | Ln 8, Col 1 | photometry_directory-Copy1.ipynb

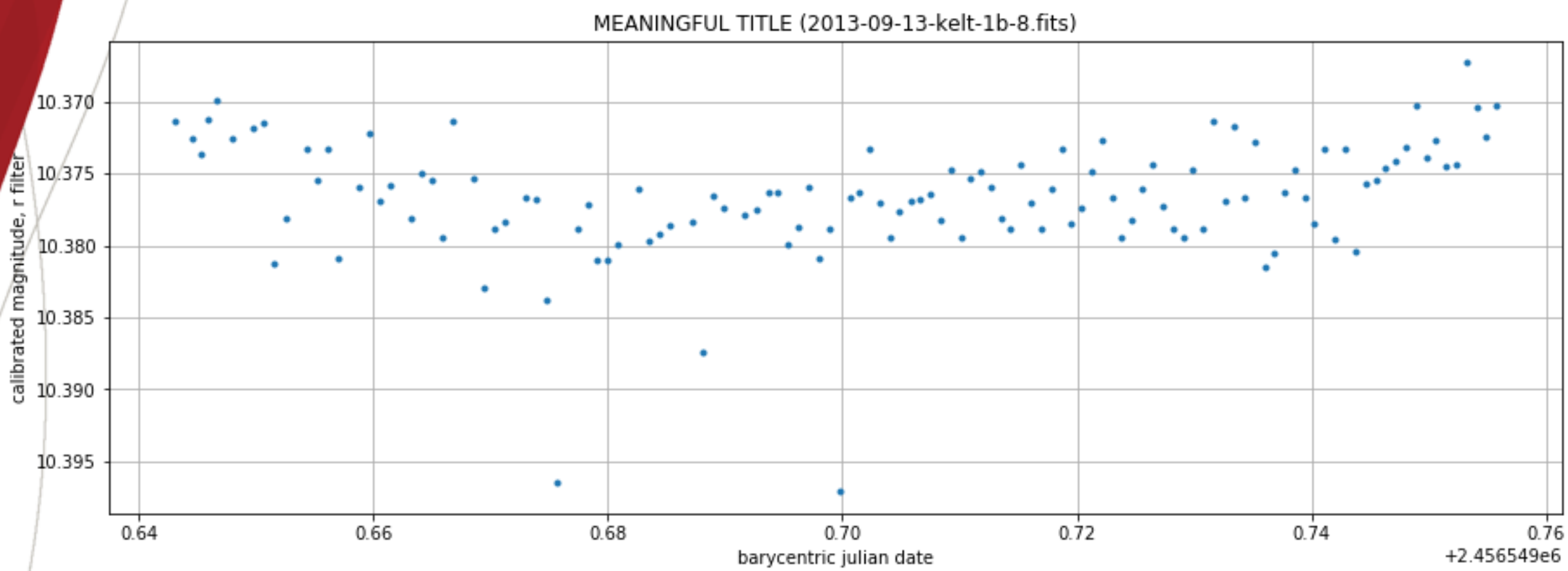
Aperture photometry

- Produces (per source per image):
  - Net counts
  - Instrumental magnitude
  - Filter
  - Sky background
  - RA/Dec
  - Error (from CCD equation)

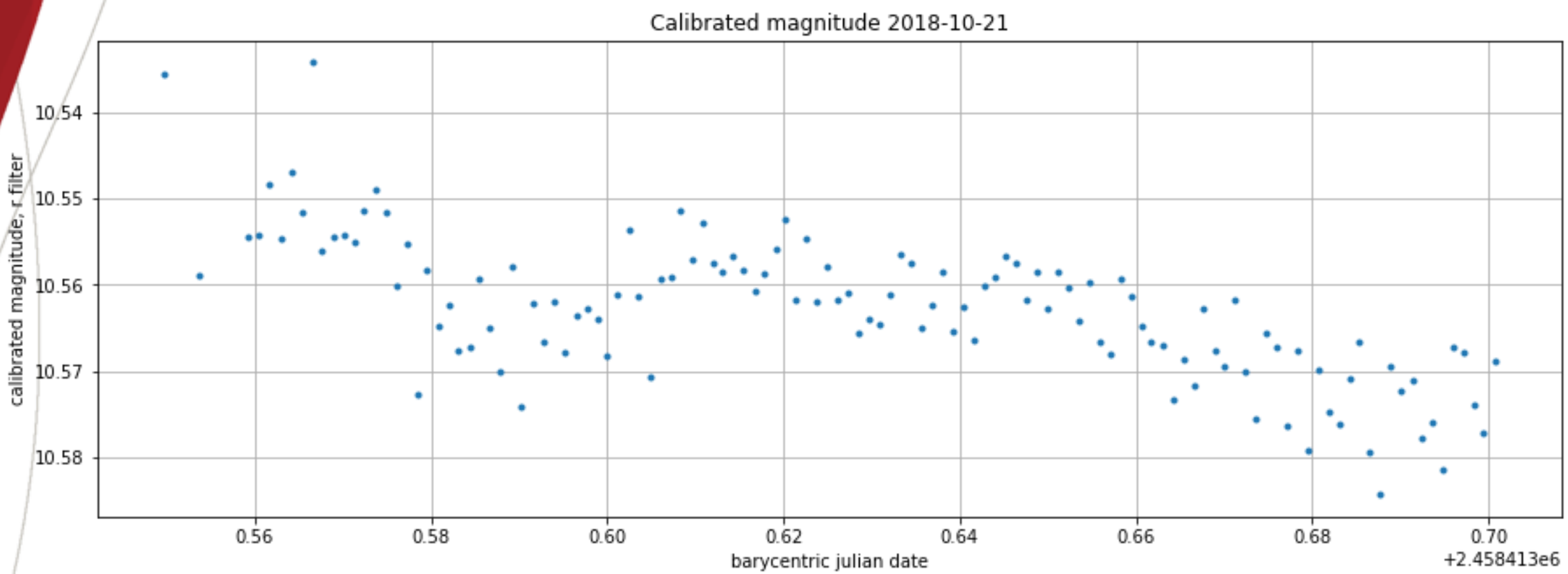
Differential photometry

- Determine color correction and zero point for each frame
  - Use APASS stars in frame as standard stars
  - Prefer those with small error
- Apply corrections to all sources

It works 😀


MEANINGFUL TITLE (2013-09-13-kelt-1b-8.fits)
calibrated magnitude, r filter
10.370
10.375
10.380
10.385
10.390
10.395
0.64
0.66
0.68
0.70
0.72
0.74
0.76
+2.456549e6
barycentric julian date

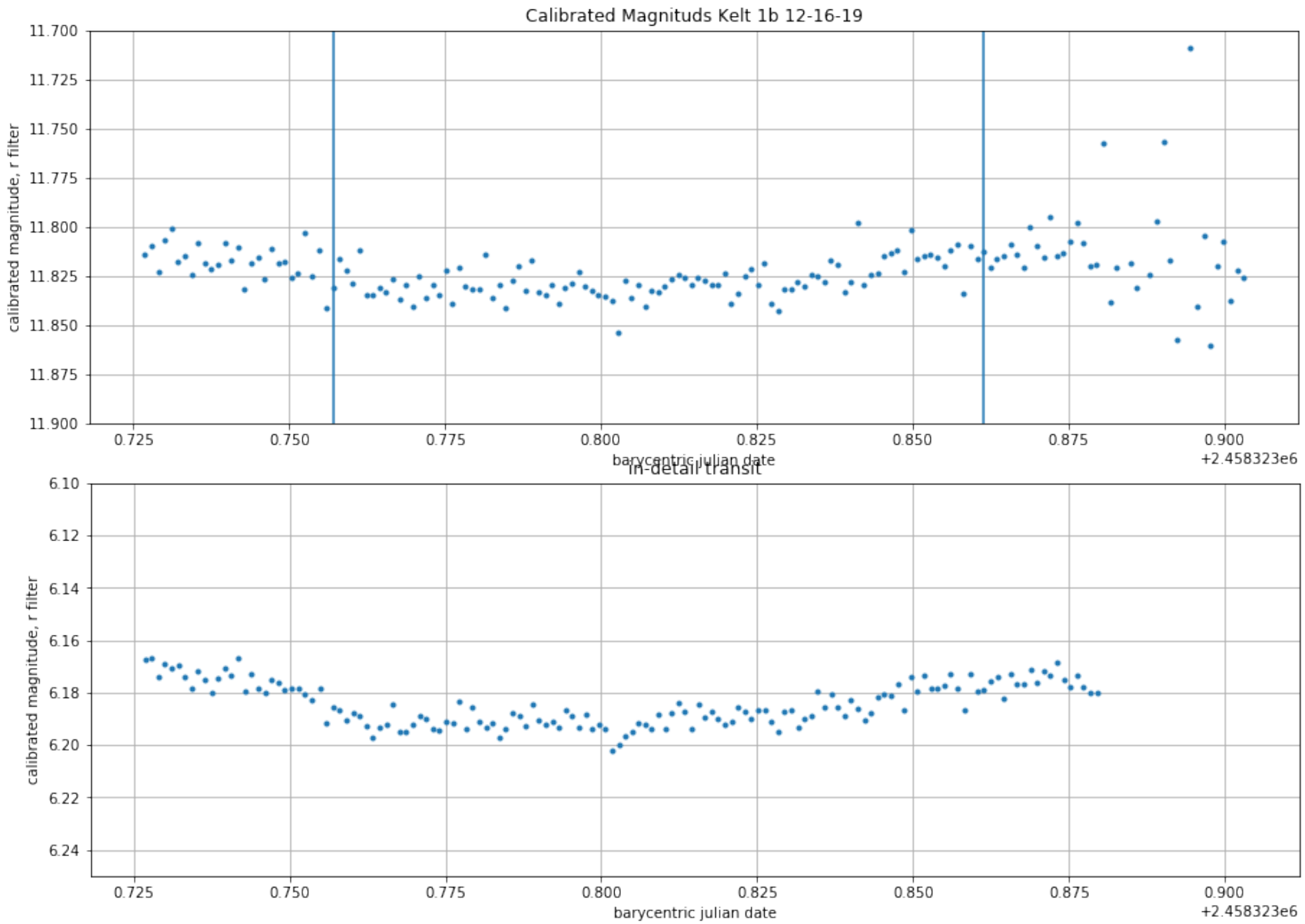
Sometimes 😬


Calibrated magnitude 2018-10-21
calibrated magnitude, r filter
10.54
10.55
10.56
10.57
10.58
0.56
0.58
0.60
0.62
0.64
0.66
0.68
0.70
+2.458413e6
barycentric julian date

Differential photometry

- Need to do differential photometry in addition
  - TESS/AIJ style
    - flux (count) ratio
    - target flux / (sum of comp fluxes)
  - AAVSO style:
    - target magnitude from
      - Difference between instrumental target and comp
      - Catalog magnitude of comp
    - average over all comp stars

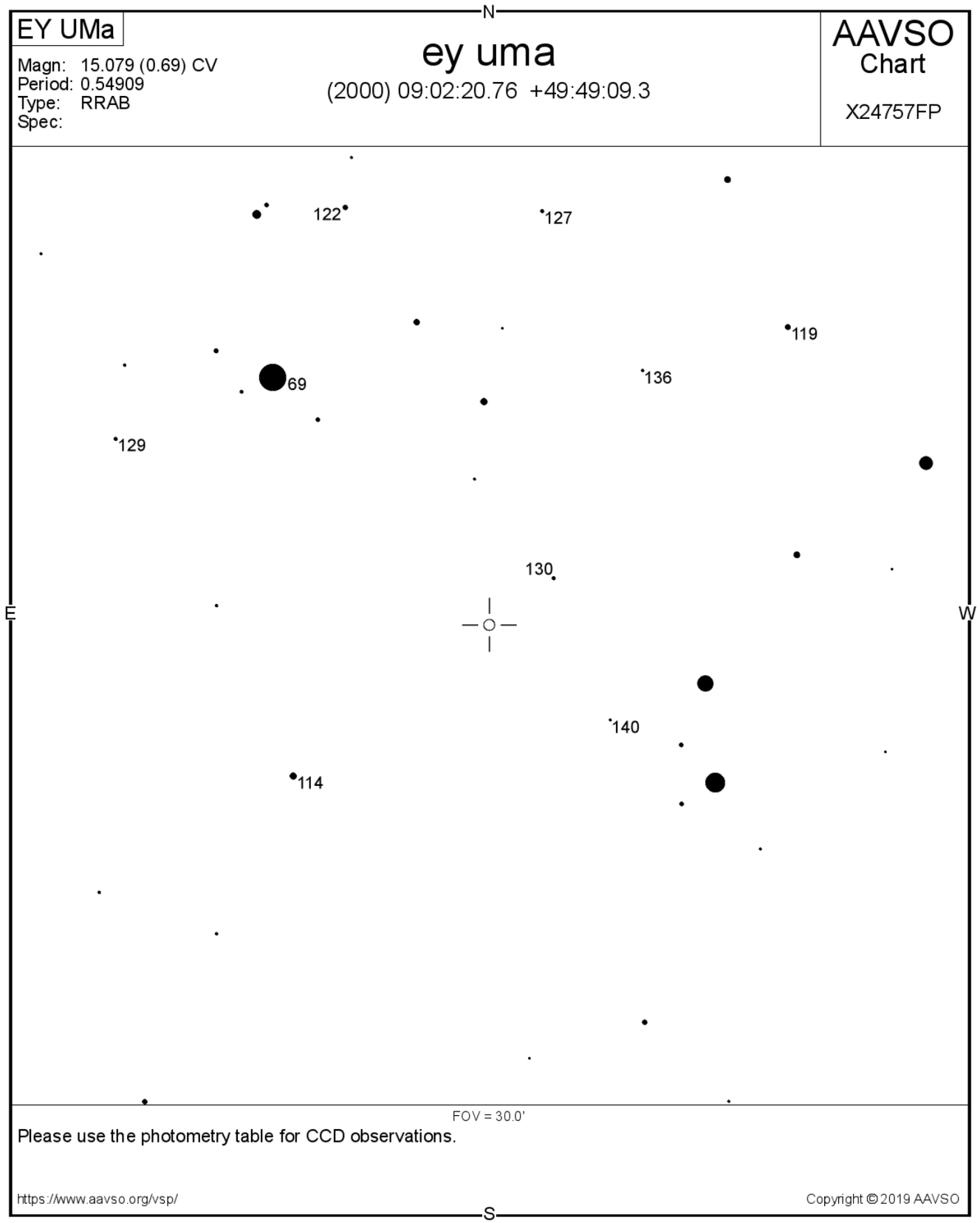
All-sky vs differential



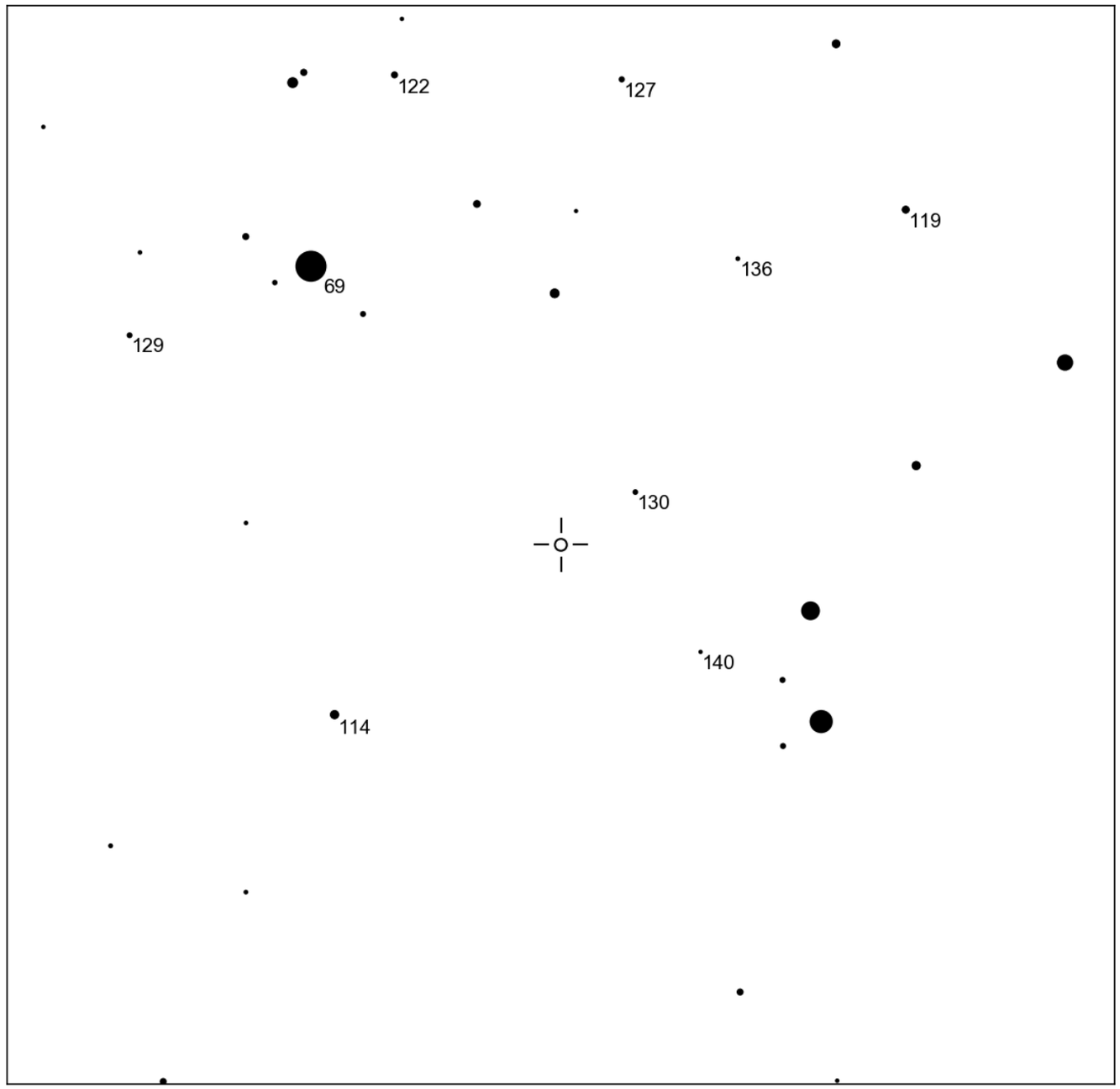
VSP

- Relatively isolated code
- Clearly defined result (single plot)

VSP: Chart for EY UMa



VSP: First pass in Python



VSP: Status

- Start: 3,500 lines of well-written Perl
- Now: 190 lines of code
  - Really poorly written Python…
  - …that uses current VSP API to get comp stars
- Eventually: 500-600 lines of code

Challenges

- Local computer
  - Installation is…painful
  - Relatively easy to break working install
  - Launch from a terminal
  - two or three platforms to support
- Server:
  - Setup is…painful
  - Authentication is…more painful
  - Storage and CPU cost money

Advantages

- Local computer
  - You control the compute
  - You have already paid for storage and CPU
- Server
  - Software for users easier to manage
  - Can provide large data files without downloads

Questions?

Slides/links/how to try it out at:

http://bit.ly/aavso-2019