Project Status

Presentation

Ruben Carcamo

.

October 6, 2024

Introduction

This presentation provides the advance of status in the class working with data from SDSS and TESS

Data from SDSS

Sloan Digital Sky Survey it's a major astronomical survey that has mapped a large portion of the sky and provided detailed data about billions of celestial objects, including galaxies, stars, and quasars.

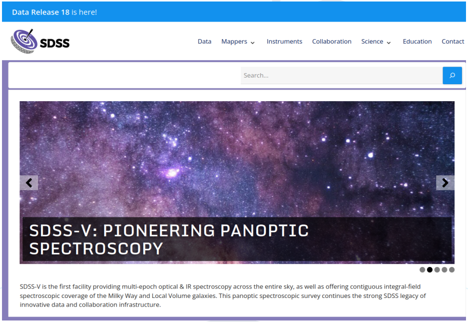


Figure: SDSS

Data from SQL (Cataclysmic Variables)

The first thing we do is using the SQL query to obtain several data like basic photometric data of 10,000 CV. The file was download in order to work with pandas.

Figure: SQL query

What we want to do?

With our csv file we have the basic information from this data the filter UGRIZ besides the RA and DEC so with this data we can plot a aintoff projection that is where is located the star in the sky in equatorial coordinates to do this task we support on jupyter notebook.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

```
In [40]: cv = pd.read_csv("10000cv.csv")
         cv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   run     10000 non-null  int64
 1   camCol  10000 non-null  int64
 2   rerun   10000 non-null  int64
 3   field   10000 non-null  int64
 4   objID   10000 non-null  int64
 5   u       10000 non-null  float64
 6   g       10000 non-null  float64
 7   r       10000 non-null  float64
 8   i       10000 non-null  float64
 9   z       10000 non-null  float64
 10  ra      10000 non-null  float64
 11  dec     10000 non-null  float64
dtypes: float64(7), int64(5)
memory usage: 937.6 KB
```

Creating a Aintoff Projection

```
In [52]: plt.subplot(projection = 'aitoff')
         plt.scatter(cv['ra'], cv['dec'], s=10, marker = '*', c='r', label = 'Cataclysmic Variables')
         plt.grid(c = 'green')
         plt.title('Aintoff projection')
         plt.xlabel('RA')
         plt.ylabel('DEC')
         plt.legend(loc =1)
         plt.savefig("aintoff.pdf")
         plt.show()
```

Figure: Jupyter notebook.

Aintoff Projection

Projection of CV

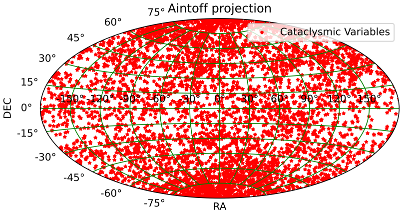


Figure: CV over sky

CMD of CV

Is possible to make an color-color diagram to see in which color are the CV.

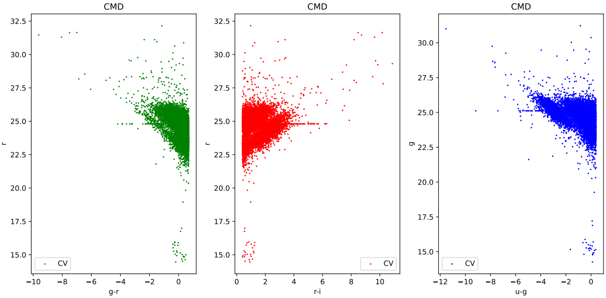


Figure: Color-Color diagram

Mean,median, max and min values

With basics statistic using pandas we get this information.

Now we calculate the mean, max, min values for each magnitude

In [190]:

```
cv.agg(
    {
        "u": ["min", "max", "median", "mean"],
        "g": ["min", "max", "median", "mean"],
        "r": ["min", "max", "median", "mean"],
        "i": ["min", "max", "median", "mean"],
        "z": ["min", "max", "median", "mean"],
    }
)
```

Out[190]:

	u	g	r	i	z
min	13.516190	14.263300	14.461610	13.547880	11.872420
max	30.377500	31.225310	32.153720	31.183070	29.717600
median	23.686765	24.813360	24.824655	23.532575	21.322145
mean	23.552356	24.668633	24.695767	23.269877	21.468740

Figure: Basic statistics with pandas.

Data from APOGEE in SDSS

There is several way to use data in SDSS the way I choose (more complicated as always do) is download the file from DR17 approx 4 GB.

Ways to Access APOGEE Data

Table of Contents

This page provides a summary of the available outputs from the different stages of the APOGEE pipelines. These can be obtained via a number of tools.

Quick Links to Data Access

We provide three primary tools to access the APOGEE data: the Science Archive Server (SAS), the Science Archive Webapp (SAW), and the Catalog Archive Server (CAS). File formats are described in the data model and we provide some examples for how to use these resources. However, many users will be satisfied with the final allStar catalog on the SAS, which gives radial velocities, stellar parameters, and abundances for each star, obtained from the combined spectra. Users should consult the "Using APOGEE Data" pages for further instructions on how to properly use and analyze the APOGEE data.

I want to ...	Tool	Tool Description	APOGEE Quick Links
Download full files to my computer and write programs to use the data.	Science Archive Server (SAS)	The SAS houses parameter summary catalogs, spectra, and all intermediate data products in flat files within a directory structure. As described in Supplemental Analyses there are multiple file trees in DR17 using different libraries. The default library is synspec with results in the synspec_rev1 filetree. (See Caveat for details)	☆ allStar File (4.0 Gb, datamodel, but see also this caveat) ☆ allStarLite (1.8 Gb, data model) ☆ allVisit File (2.8 Gb, datamodel)
Use a Web-based GUI interface to select targets or view spectra.	Science Archive Webapp (SAW)	The SAW contains spectra and associated parameters in a searchable interactive interface. The SAW is loaded from the synspec_rev1 filetree.	☆ Infrared Spectrum Search ☆ ~~Infrared Spectrum View Stars~~
Query a database with SQL to define a sample.	Catalog Archive Server (CAS)	The CAS contains the information from the summary catalogs in a queryable database structure that can be accessed through the SkyServer or directly via SQL. The CAS is loaded from the synspec filetree.	☆ IR Spec Query Form ☆ CasJobs SQL Query

Figure: APOOGE Data

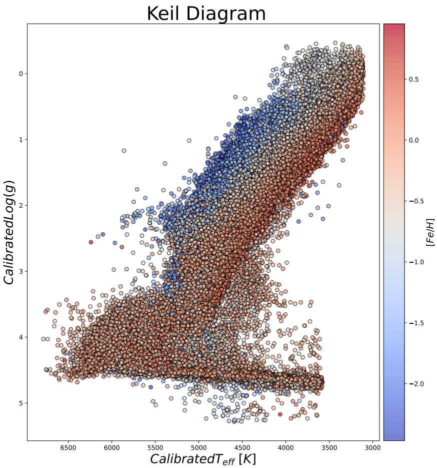
Keil Diagram

The SDSS offer a basics examples of this using python,

```
In [1]: import numpy as np
        from astropy.io import fits
        from astropy.table import Table
        import matplotlib.pyplot as plt

In [3]:
        file = './allStar-dr17-synspec_rev1.fits'
        cv = fits.open(file)

        # Refer to the data model to determine that the file has multiple HDUs and we only want the data array.
        data = cv[1].data
        # Close the remaining HDUs
        cv.close()

In [10]: # determine the integer value for the bit 23 in ASPCAPFLAG
         badbits = 2**23
         # select for ASPCAPFLAG bit 23 not being set and for EXTRATARG to have no bits set
         gd = (np.bitwise_and(data['aspcapflag'], badbits) == 0) & (data['extratarg']==0)
         #
         ind = np.where(gd)[0]

In [11]: teff_logg_check = np.logical_and(data["TEFF"] > 0, data["LOGG"] > -10) # this checks for -9999 values
         teff_logg_feh_check = np.logical_and(data["FE_H"]> -6, teff_logg_check)

         indices = np.where(np.logical_and(gd, teff_logg_feh_check))
         good = data[indices] # this only the good data now
```

Figure: Code taken from tutorial SDSS

Keil Diagram from tutorial



Data from TESS using kepler

This example wass taken from Mikulski archive in how to make light curves, in we uses python to open fits files.

In [1]:
```
from astropy.io import fits
from astropy.table import Table
import matplotlib.pyplot as plt
```

In [2]:
```
from astroquery.mast import Mast
from astroquery.mast import Observations
```

In [25]:
```
keplerObs = Observations.query_criteria(target_name='kplr011446443', obs_collection='Kepler')
keplerProds = Observations.get_product_list(keplerObs[1])
kepler_obj = Observations.filter_products(keplerProds, extension='kplr011446443-2009131110544_slc.fits',
                                          mrp_only=False)
kepler_obj
```

Out[25]: *Table masked=True length=1*

obsID	obs_collection	dataproduct_type	obs_id	description	type	
str6	str6	str10	str36	str60	str1	
601031	Kepler	timeseries	kplr011446443_sc_Q113313330333033302	Lightcurve Short Cadence (CSC) - Q0	C	mast:KEPLER/url/missions/kepler/lightcurves/0114/011446443/kpl 200913111

In [26]:
```
Observations.download_products(kepler_obj, mrp_only = False, cache = False)
```

```
Downloading URL https://mast.stsci.edu/api/v0.1/Download/file?uri=mast:KEPLER/url/missions/kepler/lightcurves/0114/
011446443/kplr011446443-2009131110544_slc.fits to ./mastDownload/Kepler/kplr011446443_sc_Q113313330333033302/kplr01
1446443-2009131110544_slc.fits ... [Done]
```

Out[26]: *Table length=1*

Local Path	Status	Message	URL
str95	str8	object	object
./mastDownload/Kepler/kplr011446443_sc_Q113313330333033302/kplr011446443-2009131110544_slc.fits	COMPLETE	None	None

In [27]:
```
kep_obj = "./kepler_obj.fits"
fits.info(kep_obj)
```

```
Filename: ./kepler_obj.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      58   ()
  1  LIGHTCURVE    1 BinTableHDU    155   14280R x 20C   [D, E, J, E, E, E, E, E, E, J, D, E, D, E, D, E, D, E, E,
E]
  2  APERTURE      1 ImageHDU        48   (8, 9)   int32
```

Figure: Jupyter notebook for TESS

Data from TESS using kepler

To use this jupyter notebook is finding and object by the KID (kepler identification object) and the notebook has basics thing in order to determine transits.

In [36]:
```
with fits.open(kep_obj) as hdulist:
    binaryext = hdulist[1].data

binarytable = Table(binaryext)
binarytable[0:5]
```

Out[36]: *Table length=5*

TIME	TIMECORR	CADENCENO	SAP_FLUX	SAP_FLUX_ERR	SAP_BKG	SAP_BKG_ERR	PDCSAP_FLUX	PDCSAP_FLUX_ERR	SAP_QUALITY
float64	float32	int32	float32	float32	float32	float32	float32	float32	int32
120.52924274596444	0.00096670596	5500	401333.16	91.52446	2598.1912	0.5752703	406145.94	130.34639	0
120.52992386784899	0.00096672785	5501	401288.16	91.51187	2598.1086	0.5752603	406100.9	127.528824	0
120.53060508973431	0.00096674974	5502	401425.53	91.53448	2598.0261	0.57525027	406242.22	125.212105	0
120.53128621167707	0.0009667717	5503	401172.0	91.517265	2597.9438	0.5752402	405984.03	123.31339	0
120.53196743356966	0.00096679357	5504	401473.62	91.53064	2597.8613	0.5752302	406293.0	121.84987	0

In [30]:
```
with fits.open(kep_obj, mode="readonly") as hdulist:
    # Read in the "BJDREF" which is the time offset of the time array.
    bjdrefi = hdulist[1].header['BJDREFI']
    bjdreff = hdulist[1].header['BJDREFF']

    # Read in the columns of data.
    times = hdulist[1].data['time']
    sap_fluxes = hdulist[1].data['SAP_FLUX']
    pdcsap_fluxes = hdulist[1].data['PDCSAP_FLUX']
```

In [31]:
```
# Convert the time array to full BJD by adding the offset back in.
bjds = times + bjdrefi + bjdreff

plt.figure(figsize=(9,4))

# Plot the time, uncorrected and corrected fluxes.
plt.plot(bjds, sap_fluxes, 'k', label='SAP Flux')
plt.plot(bjds, pdcsap_fluxes, '-b', label='PDCSAP Flux')

plt.title('Kepler Light Curve')
plt.legend()
plt.xlabel('Time (days)')
```

Figure: Jupyter notebook for TESS

Data from TESS using kepler

Plotting of the object called TRES-2

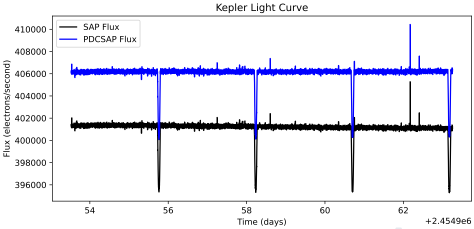


Figure: Plotting TRES-2