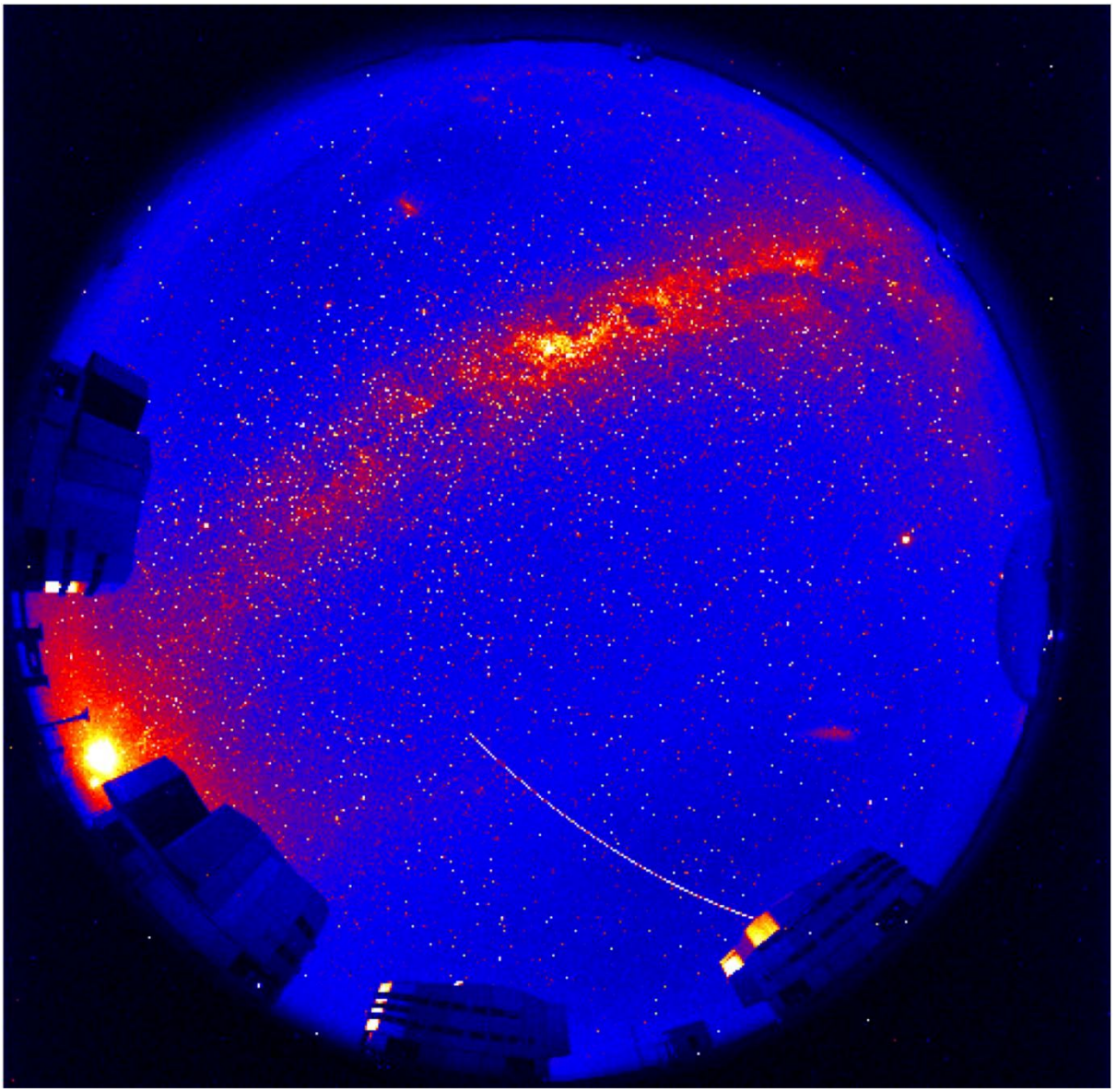
All-sky images manipulation: star registration, photometry and archive query

Context:
APICAM is an all-sky camera in Paranal, raw fits images are saved in the ESO archive and have the potential to help the weather officer by providing a map of the sky transparency.

Goals of the project:
The goal is to make the first steps in converting a raw APICAM image in a transparency map. This requires, reading a fits file, detecting automatically stars using some rejection criterias, querying Simbad to get a list of the brightest stars visible at the date of observation, and doing aperture photometry,

Steps:

1. Download and read APICAM images, use a mask to select a region of interest.
2. Detect stars in this region, reject unsuitable objects, record the position and flux.
3. Given a date and time, locate all stars brighter than magnitude V suitable for photometry.
4. If time allows, associate the stars detected on the image to the brightest stars detected from the catalogue.
5. If time allows, query the positions of the bright planets in our solar system.



ESOpy 3.0

15-16-17 April 2019

ESO Vitacura

PSF of photometric images

Project coordinated by Alessandro Razza

Abstract

Python packages for astronomers are rapidly evolving to adjust the community necessity to quickly analyze and plot astronomical data within the same framework. More than packages of a programming language, they work as authentic astronomy software providing for instance tools to detect astronomical sources and perform aperture or PSF photometry on images, as in the case of `photutils`. By implementing some features of `photutils`, `astropy` and possibly `astroquery` packages, we write a script capable to read fits files, detect stars on the images by using a model PSF and build an effective PSF (ePSF) out of the selected stars. Some optional exercise is also proposed to perform aperture photometry, to print out the list of sources with their parameters in a table and to use the stellar centroid positions, refitted with the PSF photometry routine, to evaluate the images astrometric solution. With a basic knowledge of python syntax and object-oriented programming (object instances, classes attributes and methods), working on this project simply results in finding and writing the series of instructions to perform the requested tasks.

Required background and skills

Being familiar with FITS file headers and the World Coordinate System (WCS) is required at a basic extent. Having worked with photometric data is not necessary, although it is required to have a basic knowledge of aperture and PSF photometry. No particular programming skills are required. However, understanding from ESOpy 3.0 lectures how a python object/class is instantiated and how arguments are passed to a function/class method can be beneficial.

Python packages

- `python 3.6+` and other packages used for ESOpy 3.0 (e.g. numpy, matplotlib)
- `photutils 0.6`
  Install with
  `conda install -c astropy photutils`
  from within your environment.
- `astropy 3.1`
  Install with
  `conda install -c anaconda astropy`
  from within your environment.
- `astroquery 0.3` (not strictly necessary if you do not want to query GAIA stars for the astrometry or to match the stars)
  Install with
  `conda install -c astropy astroquery`
  from within your environment.

Main project tasks

A list of the main task with a suggested sub-package to use is presented (tutoring is provided to avoid getting stuck)

1. Read a given set of photometric images (FITS files) (`astropy.io`).
2. Find stars in the images with a proper find-star algorithm (`photutils.detection`).
3. Implement a way to cut out non-stellar object (a first order solution could be matching the coordinates with GAIA stars with `astropy.coordinates` and `astroquery.vizier`)
4. build an effective PSF (ePSF) out of the selected stars following (look here to avoid wasting a month doing it!)

Optional tasks

Although recommended to complete all of the proposed optional tasks, you can pick up only one or more from the following list:

5. Compute the aperture photometry (few passages more) and PSF photometry (it is an automatic outcome of the process for finding the ePSF) of the list of stars
6. Print on browser/save in a text file the list of stars with their parameters (pixel positions, coordinates, aperture photometry, PSF photometry)
7. Save the FWHM from the ePSF in the image header
8. Evaluate the WCS solution by comparing the PSF centroid positions of the stars (it is an automatic outcome of the process for finding the ePSF) with GAIA catalog

ESOpy3.0

15-16-17 April 2019

Afternoon Project:

Simple spectral stacking

R. Thomas

Required Packages: argparse, numpy, catscii

Aim:

This projects aims at creating a code that stack (average) spectra of galaxies together.

Material provided:

-We provide for this project 400 (public :)) spectra of galaxies at 0.8<z>1.0 in ascii format.

Final product:

The goal is to make a software (not a module).

-We will create a command line interface [CLI] that takes 2 mandatory argument:

- The list of spectrum-redshift. Other optional arguments are
- A place where to normalize the spectra

Few optional arguments can also be included such as the binning of the stacked spectrum, the name of the final spectrum, etc...this is up to you.

-Once you design the CLI, you must work on each individual spectrum from the list: de-redshift them and normalize them. You must save them all together and then interpolate them on a common wavelength grid.

-Then you will have to combined them. The way to combine them is up to you. You might want to just take the average of all the spectra or you can also use a sigma-clipping algorithm before taking the mean..

-Finally, you will save the spectrum in a text file with
-lambda, flux, standard deviation-

Example of CLI:

```
usage: specstack [-h] [-s S] [-p] [-f F] speclist normlimits bin

specstack V19.4.0, R. Thomas, 2018, ESO, This program comes with ABSOLUTELY NO
WARRANTY; and is distributed under the GPLv3.0 Licence terms.See the version
of this Licence distributed along this code for details.

positional arguments:
  speclist    File with col1 = spectra names, col2 = redshift
  normlimits  l1 and l2 in angstrom where the spectra will be normalised
  bin         Binning of the stacked spectrum

optional arguments:
  -h, --help  show this help message and exit
  -s S        Sigma we use for the clipping, default=3
  -p          If plot at the end
  -f F        Name of the final file
```

Celestial object visibility plots

Elyar Sedaghati

Level: Beginner → Intermediate

Package requirement: `numpy, matplotlib, astropy`

January 7, 2018

In this exercise we will write a simple code for coordinate transformations and the creation of visibility curves. It is planned to demonstrate the capabilities of the `astropy.coordinates` package. For more convenient and/or complex observation planning, you should consider the `astroplan` package.

Here are some hints that should help with designing the code. Please click on the coloured links for the documentation of that specific class/package.

- Import the basic necessary packages (numpy, matplotlib)
- Import packages for finding and transforming coordinates (look into the `astropy` package)
- The variables should ideally be soft-coded, where the user is asked to input the target name, the date of the observations and the observatory name.
- Provide the user with a list of available observatories to choose from in `astropy.coordinates.EarthLocation`
- Get the coordinates of the the target provided by the user (`astropy.coordinates.SkyCoord`)
- Get the coordinates of the the telescope provided by the user (`astropy.coordinates.EarthLocation`)
- Calculate alt, az coordinates of the target for the given observatory (`astropy.coordinates.AltAz` & the `transform_to` method in the `SkyCoord` class)
- Convert alt, az to airmass and plot it as a function of time (`secz` method in the `AltAz` class)
- Calculate alt, az values for both the sun and the moon during the night of the observations (`astropy.coordinates.get_sun` and `astropy.coordinates.get_moon`)
- Remember to recalculate alt, az coordinates of the target for these new time stamps.
- Make the final visibility plot with `matplotlib`, which includes the altitude of the target, the moon and the sun, indicates twilight as gray-shaded and dark time as black-shaded. Optionally you can include the azimuth variations of the target as a `colorbar`. For a nicer-looking plot, I recommend styling the `matplotlib` graphs with the `astropy_mpl_style` method from the `astropy.visualization` package.

Title: Tapping into the ADS with Python

Concepts involved: manipulation of lists, 2D plots (line, scatter, histogram), basic data fitting

Abstract:

The astroquery module allows to easily query the ADS with Python. Here, I propose to use this module to assemble a simple script that computes one's publication statistics automatically. The tasks involved (list manipulation, basic plotting) are ideal for beginners and intermediate users to develop their programming skills. Advanced users may use the datasets involved to explore a range of data manipulation and fitting routines. The goal is for every participant to go home with a fully working script able to automatically update their publication statistics.

Packages required:
matplotlib, astroquery, numpy

Packages possibly useful:
dateutil, datetime, scipy, astropy, statsmodels

ESOpy3.0

15-16-17 April 2019

Afternoon Project:

catmatch

R. Thomas

Required Packages: numpy, catscii, tqdm (optional)

Aim:

This project aims at matching catalogs by column-row entries.

Material provided:

Two catalog with a common column

Final product:

The goal is to make a software (not a module).

-We will create a command line interface [CLI, see an example below] that takes 4 mandatory argument:

- The two catalog two match
- The name of the column to be matched.
- The name of the final file

-Once you design the CLI, you must must take the catalogs and read them.

-Then look for the column you want to match and start to loop over the first catalog rows.

-For each row you will take the column to match and look into the other catalog to find a match

-Once it is done for each row, you will have to write down the matched combined catalog (see below) with the proper header. If header columns are the same you should also take this into account.

- You must also think about what happens when in the second catalog, to rows have the same ID for the matching column, or when there is nothing to match with.
---In brief, you must go from two catalogs like these ones:

Cat1:

ID	Col1	Col2	Col3	....
ID#1	X1	Y1	Z1	....
ID#2	X2	Y3	Z2	....
ID#3	X3	Y3	Z3	....
ID#4	X4	Y4	Z4	....
ID#5	X5	Y5	Z5	....

And Cat2:

ID	Col1	ColZ	ColX	....
ID#1	A1	B1	C1	....
ID#2	A2	B3	C2	....
ID#3	A3	B3	C3	....
ID#4	A4	B4	C4	....
ID#5	A5	B5	C5	....

and end up with:

ID	Col1	Col2	Col3	....	ID	Col1_1	ColB	ColC	....
ID#1	X1	Y1	Z1	....	ID#1	A1	B2	C2	....
ID#2	X2	Y3	Z2	....	ID#2	A2	B2	C2	....
ID#3	X3	Y3	Z3	....	ID#3	A3	B3	C3	....
ID#4	X4	Y4	Z4	....	ID#4	A4	B4	C4	....
ID#5	X5	Y5	Z5	....	ID#5	A5	B5	C5	....

Example of CLI:

usage: catmatch [-h] file1 file2 column outputfile

specstack V1.2, R. Thomas, 2018, ESO, This program comes with ABSOLUTELY NO WARRANTY; and is distributed under the GPLv3.0 Licence terms.See the version of this Licence distributed along this code for details.

positional arguments:
  file1   Your first catalog of data to match this is mandatory
              (positional argument)
  file2   Your second catalog of data to match this is mandatory
              (positional argument)
  column   The common column to match. The word you enter here must be in
              the header of the column you want to match and must be in the
              two catalogs
  outputfile The name of the output file that will be created (without
              spaces)

optional arguments:
  -h, --help show this help message and exit

ESOpy3.0

15-16-17 April 2019

Afternoon Project:

dfitspy

R. Thomas

Required Packages: fitsio, numpy [& python-magic]

Aim:

This project aims at porting the dfits|fitsort algorithms to python

Material provided:

-5 test FITS files.

-FITS files search function (digging into the fitsio C-wrapper)

-File search function

Final product:

The goal is to make a software (not a module).

-We will create a command line interface [CLI, see an example below] that takes 2 mandatory argument:

- The list of files
- The list of keywords

Few optional arguments can also be included such as grepping value, keyword listing, saving the results, etc…. This is up to you!

-Once you design the CLI, you must work on each individual file and look for the keywords you are asking to display. When you find them you must save the file, the keyword, and the corresponding value.

-Then you must display them in a *dfits* fashion (see below)

Example of CLI:

```
usage: dfitspy [-h] [-file [FILE [FILE ...]]] [-key KEY] [--list]
               [--grep GREP] [--save] [--test] [--version] [--docs]

dfitspy: dfits|fitsort in python, version 19.3.4, Licence: GPL

optional arguments:
  -h, --help          show this help message and exit
  --list              List all keywords in a given file (if a list of file
                      is given the first one is used)
  --grep GREP         Restrain the files to the one with a given value of a
                      given parameter. It can be used multiple times with
                      different values
  --save              Save the list of files into an ascii file
  --test              Start the testing of the program
  --version           Display the version of the program
  --docs              Diplay the online or local documentation program

Mandatory arguments if you want to dfitsort your files:
  -file [FILE [FILE ...]]
                      a file, a list of file separated by coma, *.fits is
                      accepted, * as well, test* as well, testdir/test* as
                      well
  -key KEY            Header keyword or list of header keywords (separated
                      by coma)
```

Example of output:

```
[DFITSPY INFO]> 34 fits files will be considered

filename                                                OBJECT          LST             ESO OBS ID
------------------------------------------------------  ------------    ---------       ----------
r.XSHOO.2099-14-59T09:53:43.577_tpl-A01_0000.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:43.577_tpl-A01_0001.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:43.577_tpl-A01_0002.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:43.577_tpl-A02_0000.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:43.577_tpl-A02_0001.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:43.577_tpl-A02_0002.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:43.577_tpl-A03_0000.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:43.577_tpl-A03_0001.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:43.577_tpl-A03_0002.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:44.797_tpl-A01_0000.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:44.797_tpl-A01_0001.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:44.797_tpl-A01_0002.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:44.797_tpl-A02_0000.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:44.797_tpl-A02_0001.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:44.797_tpl-A02_0002.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:44.797_tpl-A03_0000.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:44.797_tpl-A03_0001.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:44.797_tpl-A03_0002.fits       LAMP,AFC        78684.245       2025011
r.XSHOO.2099-14-59T09:53:45.055_tpl-A01_0000.fits       LAMP,AFC        78685.247       2025011
r.XSHOO.2099-14-59T09:53:45.055_tpl-A01_0001.fits       LAMP,AFC        78685.247       2025011
r.XSHOO.2099-14-59T09:53:45.055_tpl-A01_0002.fits       LAMP,AFC        78685.247       2025011
r.XSHOO.2099-14-59T09:53:45.055_tpl-A02_0000.fits       LAMP,AFC        78685.247       2025011
r.XSHOO.2099-14-59T09:53:45.055_tpl-A02_0001.fits       LAMP,AFC        78685.247       2025011
r.XSHOO.2099-14-59T09:53:45.055_tpl-A02_0002.fits       LAMP,AFC        78685.247       2025011
r.XSHOO.2099-14-59T09:53:45.055_tpl-A03_0000.fits       LAMP,AFC        78685.247       2025011
r.XSHOO.2099-14-59T09:53:45.055_tpl-A03_0001.fits       LAMP,AFC        78685.247       2025011
r.XSHOO.2099-14-59T09:53:45.055_tpl-A03_0002.fits       LAMP,AFC        78685.247       2025011
r.XSHOO.2099-14-59T09:59:57.509_tpl-A01_0000.fits       STD,TELLURIC    79056.26        2025011
r.XSHOO.2099-14-59T09:59:57.509_tpl-A01_0001.fits       STD,TELLURIC    79056.26        2025011
r.XSHOO.2099-14-59T09:59:57.509_tpl-A01_0002.fits       STD,TELLURIC    79056.26        2025011
r.XSHOO.2099-14-59T09:59:57.509_tpl-A01_0003.fits       STD,TELLURIC    79056.26        2025011
r.XSHOO.2099-14-59T09:59:57.509_tpl-A01_0004.fits       STD,TELLURIC    79056.26        2025011
r.XSHOO.2099-14-59T09:59:57.509_tpl-A01_0005.fits       STD,TELLURIC    79056.26        2025011
r.XSHOO.2099-14-59T09:59:57.509_tpl-A01_0006.fits       HD 205828       79056.26        2025011

[DFITSPY INFO]> 34 files used in output
```