Lab_1_Introduction

March 4, 2022

1 Lab 1 Introduction to Jupyter Notebooks & FITS images

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. In this notebook, you will be introduced to Jupyter Notebook, learn the basics tools, open a FITS image and perform photometry.

By the end of this lab, you should be able to: - Use the different cells types - Load in the appropriate python libraries need to complete your task - Read in a FITS file and display the image in both ds9 & in a Jupyter Notebook - Understand the structure of the lab and the marking rubric

1.1 In a notebook, there are two types of cells:

- A **code cell** which contains code to be executed in the kernel and displays its output below.
- A **Markdown cell** which contains text formatted using Markdown and displays its output in-place when it is run

1.2 Markdown Cells

Cell one and two in this notebook are Markdown cells. You will use Markdown cells to take notes, answer questions and write observations and conclusions, so it is important that you are comfortable with using Markdown cells by the end of this lab. You can "code" your Markdown cell to bold text, use different colours and write equations similar to LaTeX. Once you are done writing out a Markdown cell, you run it by clicking the Run button on the Menu bar or using the keyboard short-cut "Ctrl+Enter."

1.2.1 Below are a few basic tasks to help you become comfortable with running cells, adding cells and editing markdown.

Task 1. If the 1st and 2nd cells in this notebook has not been run, run them.

Task 2. Add a Markdown cell below. You can do this by adding a new cell (Insert- Insert Cell Below) and then changing the type to Markdown or using a keyboard shortcut.

(Here is an extended list of Jupyter Notebook Short cut: https://towardsdatascience.com/jypyter-notebook-shortcuts-bf0101a98330

Task 3. In the Markdown cell, introduce yourself to the TAs. Please mention your major, why did you choose to take this unit, what you hope to learn in the unit, any fun facts you want to tell us and (most importantly) your favorite cookie. In this Markdown cell, you need make sure, you use: - two types of headings - different colours. Do not use black for your responses (**for everything**). It helps the TAs distinguish your responses. - *either bold or italic font*

(Here is Markdown cell cheatsheet to help you out: https://medium.com/ibm-data-science-experience/markdown-for-jupyter-notebooks-cheatsheet-386c05aeebed)

2 Hi!

2.1 Hello!

My name is Cedric, I'm a final year Astro and Computer Science student here. I chose this unit because it'll give me practical experience in working on **real data sets** and bridge the gap between theory and practice.

3 Code Cells

Code Cells basically allows you to enter and run code. You can import libraries, write functions and objects, plot data, etc. Throughout this unit, we will use code cells to process, reduce and visualize FITS images. We will provide a **(short)** pre-lab coding practice problems as well as science questions each week and extra resources so by the end of the semester you will be comfortable with coding and Jupyter Notebook

3.1 Let begin this weeks notebook:

Task 4. As a first step, we need to import the python libraries that are critical for today's workshop, including numpy (numerical), astropy (astronomy), ccdproc (CCD) and matplotlib (plotting). Click on the cell below and then click the run button above. (Don't forget to run cell too if you haven't).

[ ]:
```
import numpy as np
import astropy
import ccdproc
from ccdproc import CCDData, combiner
from astropy import units as u
import matplotlib.pyplot as plt
```

4 Python Libraries:

One of the many reasons why we are using python is the amount of documentation and help guides that are available via a basic google search. **You should very careful about what site you use** but you can find a tutorial or a guide for just about anything you are trying to do.

Task 5. As you see above, we used different import statements. What does each different import statement mean/do?

I.E. how does *import [library]* diff from *import [library] as [shortname]*

Task 6. In a new Markdown box, write a 1 sentence descripition about each of the library we are importing.

(Here is a list of reasons why you should Python: https://medium.com/@mindfiresolutions.usa/python-7-important-reasons-why-you-should-use-python-5801a98a0d0b)

Task 5: Import without any qualifiers imports all classes and functions from the called file/package into the current file, but they are not added to the global namespace. Using their functions requires the qualification of the name of the package first.

Importing *as* allows for the assignment of an alias to the above imports, but still must be called explicitly with (for example) np.array().

Using the *from* keyword is riskier, since it imports the selected classes/functions directly into the global (top-level) namespace. These can be called directly (ie, CCData(*args)).

Task 6: - numpy is a general library for computational mathematics, which is implemented in C to be more performant than base Python - astropy is a library for common astronomical operations - ccdproc is a library that provides ways to reduce astronomical images - matplotlib is a library that provides functionality for generating graphs of various sorts

Task 7. Lets now load an example FITS image from the Monash C14 telescope, NGC_2362_V_30.000secs00010838.fit.

[ ]:
```
image_1 = CCDData.read("NGC_2362_V_5.000secs00009644.fit", unit="adu")
```

```
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 58206.525361
from DATE-OBS'. [astropy.wcs.wcs]
```

Task 8. Lets now print some example pixel values from this image. Display this image with ds9 too, and compare the individual pixel values (starting at the corners of the image). Record your observations in a new Markdown cell. For example, which pixel values correspond to which corner of the image you loaded into ds9.

[ ]:
```
print(image_1)
```

```
[[ 999 1031 1039 … 1041 1001 1033]
 [1066 1068 1035 …  983 1005 1043]
 [1065 1062 1091 … 1047 1002 1040]
 …
 [1052 1019 1025 … 1033 1024 1062]
 [1077 1077 1098 … 1022 1052 1042]
 [1074 1046 1079 … 1078 1047 1005]] adu
```

image_1[0,0] refers to the bottom left corner

image_1[-1,0] refers to the top left corner

image_1[-1,-1] refers to the top right corner

image_1[0,-1] refers to the bottom right corner

The first index refers to the vertical axis and the second index refers to the horizontal axis. The origin is located in the bottom left hand corner.

Task 9. Lets print just one specific pixel value. Where is this pixel value in the image that is displayed with ds9?

[ ]:
```
print(image_1[0,0])
```

```
999 adu
```

Task 10. Lets print a specific row. Confirm where this row is in your image using ds9.

```
[ ]: print(image_1[0])
```

```
[ 999 1031 1039 … 1041 1001 1033] adu
```

Task 11. Lets print a specific column. Confirm where this column is in your image using ds9.

```
[ ]: print(image_1[:,0])
```

```
[ 999 1066 1065 … 1052 1077 1074] adu
```

Task 12. How would you print a pixel value (or values) at the location of the bright star near the centre of the image? Use ds9 to assist you and then write the relevant command below in a code cell and run it.

```
[ ]: print(image_1[1270, 1713])
```

```
65535 adu
```

Task 13. We can now use numpy (np) to determine some of the basic statistics of the image, and print to display them to our screen.

```
[ ]: print('Min:', np.min(image_1))
     print('Max:', np.max(image_1))
```

```
Min: 577
Max: 65535
```

Task 14. How would you determine the mean and standard deviation of the image? There are online resources that can help you determine which functions to use.

```
[ ]: print('Mean:', np.mean(image_1))
     print('StdDev:',np.std(image_1))
```

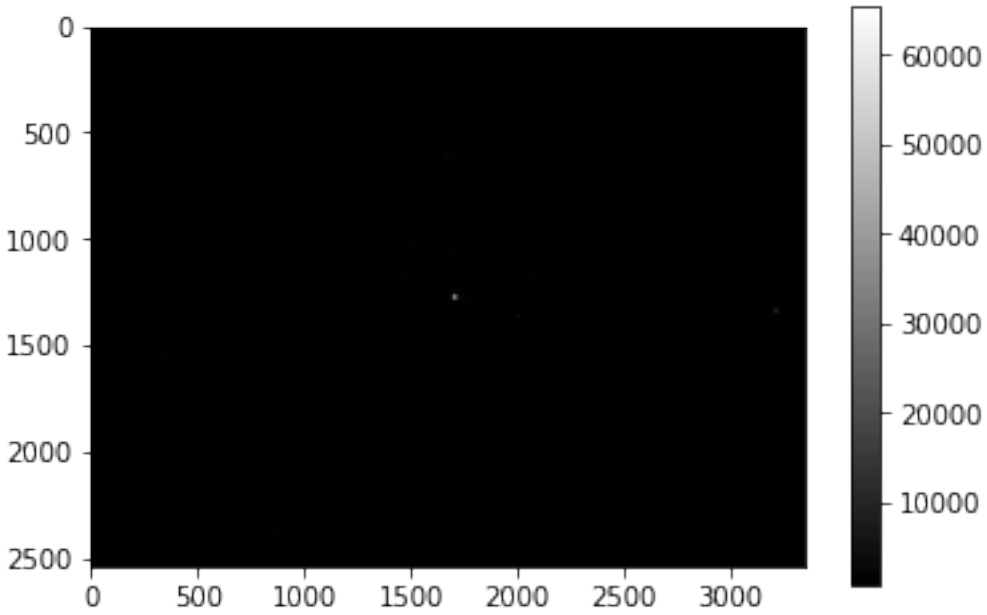
```
Mean: 1031.0610487127121
StdDev: 331.65049691007994
```

5 Reminder: are you keeping notes as you work through this notebook? For each Task, you should have written notes/ observations in a new markdown cell.

Task 15. Lets now display the image using matplotlib and compare what we see with ds9.

```
[ ]: plt.imshow(image_1, cmap='gray')
     plt.colorbar()
```

```
[ ]: <matplotlib.colorbar.Colorbar at 0x169c14e20>
```



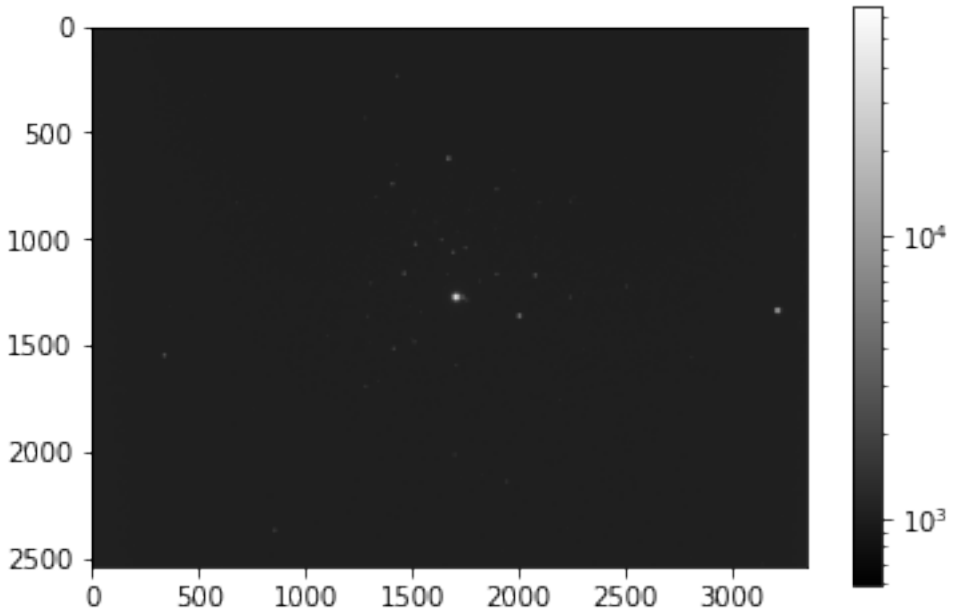
5.0.1 As you can see, the image is basically featureless. This is due to the colourbar that we have selected. In ds9, we can change the colourbar and edit the min and max value until we are happy with the images. We can try the same here.

Task 16. Lets now try a different colour scale, so we can see more stars in the image. In this case, let's try the log-normalisation option, which we need to import from matplotlib.colors. Comment on difference in the image (i.e. how many more stars do you see, where are the 2 stars located? Compare both the numerical pixel coordinates of the 2 stars in this notebook to ds9? How do coordinates differ?)

[ ]:
```
from matplotlib.colors import LogNorm
```

[ ]:
```
plt.imshow(image_1, cmap='gray', norm=LogNorm())
plt.colorbar()
```

[ ]:
```
<matplotlib.colorbar.Colorbar at 0x169d5dc30>
```



The first index in the notebook refers to the y coordinate in ds9. The second index in the notebook refers to the x coordinate in ds9.

6 Continue from here

6.0.1 Looking at images of stars is nice, but we want to quantify the properties of those stars.

Task 16. To do this, lets undertake photometry using the photutils libraries. Write a 1 sentence descriptions of each of the modules we are importing in.

[ ]:
```
from photutils import CircularAperture
from photutils import aperture_photometry
from photutils import Background2D
from photutils import MedianBackground
```

`CircularAperture` is a class that represents a circular aperture in pixel coordinates.

`aperture_photometry` is a function that performs aperture photometry on an Aperture item and a given dataset.

`Background2D` is a class that provides methods to estimate the 2D background and associated RMS noise in an image.

`MedianBackground` calculates the median background given a set of data and sigma-clipping parameters

If we are to measure photometry, we need to make sure we subtract off the photons (electrons) coming from the background sky rather than celestial sources.

Task 17. We can create a background image using Background2D. What is Background2D doing? Take a look at https://photutils.readthedocs.io/en/stable/api/photutils.background.Background2D.html#photutils.background. to find out. What are the roles of each of the parameters that we have defined?

```
[ ]: bkg = Background2D(image_1, box_size=100, bkg_estimator = MedianBackground() )
```

The first arg passes in the image's adu data array. The second argument defines discrete boxes/data chunks for which to use to process the data. The last argument (note that these need to be called using the keywords) defines the method/algorithm that is used for this calculation.

Task 18 Lets now do print(bkg) and see what comes out.

```
[ ]: print(bkg)
```

```
<photutils.background.background_2d.Background2D object at 0x16a9b5c30>
```

A reference to the object is printed, as no useful `__str__` method has been defined for the class.

Task 19. Lets now print the attributes of the bkg class.

```
[ ]: print(dir(bkg))
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__', '__weakref__', '_bkg_stats', '_bkgrms_stats',
'_box_data', '_box_idx', '_box_npixels_threshold', '_combine_masks',
'_filter_meshes', '_get_box_indices', '_interpolate_meshes', '_make_2d_array',
'_make_mesh_image', '_mesh_idx', '_mesh_xypos', '_mesh_yxpos',
'_prepare_box_data', '_prepare_data', '_process_size_input', '_reshape_data',
'_select_initial_boxes', '_selective_filter', '_sigmaclip_boxes',
'_unfiltered_background_mesh', '_validate_array', 'background',
'background_median', 'background_mesh', 'background_mesh_ma',
'background_mesh_masked', 'background_rms', 'background_rms_median',
'background_rms_mesh', 'background_rms_mesh_ma', 'background_rms_mesh_masked',
'bkg_estimator', 'bkgrms_estimator', 'box_npixels', 'box_size', 'coverage_mask',
'data', 'edge_method', 'exclude_percentile', 'fill_value', 'filter_size',
'filter_threshold', 'interpolator', 'mask', 'mesh_nmasked', 'nboxes',
'nboxes_tot', 'plot_meshes', 'sigma_clip', 'total_mask', 'unit']
```

Task 20. Lets now look at the value of one of these attributes. What information is this attribute outputting?

```
[ ]: print(bkg.background_median)
```

```
1027.0 adu
```

This calculates the median background ADU across all pixels in the image file.

Task 21. Print out the number of pixels in bkg using an attribute

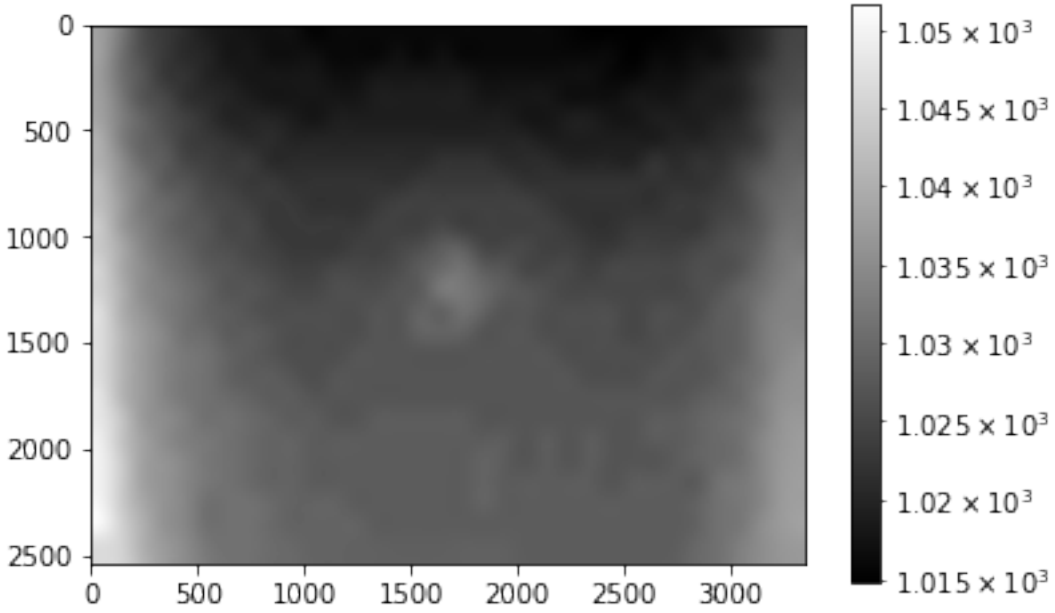
```
[ ]: print(bkg.background.size)
```

```
8515888
```

Task 21. Now lets take a look at the background image.

```
[ ]: plt.imshow(bkg.background, cmap='gray', norm=LogNorm())
     plt.colorbar()
```

```
[ ]: <matplotlib.colorbar.Colorbar at 0x16ba828c0>
```



Task 22. Below are the commands that undertake aperture photometry of two stars. Can you describe what each line does, and what the relevant inputs are?

```
[ ]: positions = [(1713.0, 1274.0), (3220.0, 1337.0)]
     apertures = CircularAperture(positions, r=20.0)
     phot_table = aperture_photometry(image_1-bkg.background, apertures)
```

The first line defines tuples of the y-x coordinates of the centre of the stars. The second line defines two circular apertures at the locations with a radius of 20 pixels. The third line performs the aperture photometry.

Task 23. We can now print the photometry to the screen.

```
[ ]: print(phot_table)
```

```
 id xcenter ycenter    aperture_sum
      pix     pix          adu
--- ------- ------- ------------------
  1  1713.0  1274.0 17778817.173594087
  2  3220.0  1337.0 4177791.9190476146
```

Task 24. Now undertake aperture photometry of four stars with an aperture radius of 15 pixels.

[ ]:
```
print(aperture_photometry(image_1-bkg.background,␣
↪CircularAperture([positions[0], positions[1], (683, 834), (1287, 437)],r=15.
↪0)))
```

```
 id xcenter ycenter    aperture_sum
      pix     pix          adu
--- ------- ------- -----------------
  1  1713.0  1274.0 16956870.10063022
  2  3220.0  1337.0 4052048.313483941
  3   683.0   834.0 43109.53842358233
  4  1287.0   437.0 64423.89031254575
```

7 Lab Conclusion:

You have now completed the "coding" part of the lab. Before you submit your notebook to be marked, here is a quick checklist you should go through to ensure you have fully completed the lab and achieved the learning goals stated in the first cell of the lab.

```
1. Did I write a conclusion?

2. Did I complete all the **Tasks**? Tasks will be a mix of running code, answering questions a

3. Does my Jupyter notebook run? The TAs will open each notebook and press "run all" to check i

4. Do I understand every bit of this Notebook? Each lab, we will be building upon what we did i
```

8 Pre-Lab Questions:

Each Monday, a pre-lab jupyter notebook will be posted which you will need to complete and sumbit (via Moodle) before Thursday 2pm. The notebook will either have a short coding exercise, concept questions or both. The purpose of the pre-labs are provide background for that week's lab and hand-on practice with python, so you can preform well in lab. It is very important that you complete each pre-lab, as it will also contribute to your lab mark for that week.