Exploring Astropy (FITS handling) and learn DS9 tool

Astropy

Astropy is a Python package for astronomy that provides core functionality and common tools for astronomy and astrophysics.

You can explore it here: https://docs.astropy.org/en/stable/index_user_docs.html

In [6]:
```
pip install astropy
```

```
Requirement already satisfied: astropy in /opt/anaconda3/lib/python3.11/site-packages (5.3.4)
Requirement already satisfied: numpy<2,>=1.21 in /opt/anaconda3/lib/python3.11/site-packages (from astropy) (1.2
6.4)
Requirement already satisfied: pyerfa>=2.0 in /opt/anaconda3/lib/python3.11/site-packages (from astropy) (2.0.0)
Requirement already satisfied: PyYAML>=3.13 in /opt/anaconda3/lib/python3.11/site-packages (from astropy) (6.0.1
)
Requirement already satisfied: packaging>=19.0 in /opt/anaconda3/lib/python3.11/site-packages (from astropy) (23
.1)
Note: you may need to restart the kernel to use updated packages.
```

In [12]:
```
#To install astropy, run:
#pip install astropy

import astropy
print(f"Astropy version: {astropy.__version__}")  # You can check astropy version
```

```
Astropy version: 5.3.4
```

Constants in Astropy

Astropy provides physical and astronomical constants with units.

In [16]:
```
from astropy import constants as const

# Fundamental constants
print(f"Speed of light: {const.c}")
print(f"Gravitational constant: {const.G}")
print(f"Planck constant: {const.h}")

# Astronomical constants
print(f"Solar mass: {const.M_sun}")
print(f"Earth mass: {const.M_earth}")
print(f"Parsec: {const.pc}")
```

```
Speed of light: 299792458.0 m / s
Gravitational constant: 6.6743e-11 m3 / (kg s2)
Planck constant: 6.62607015e-34 J s
Solar mass: 1.988409870698051e+30 kg
Earth mass: 5.972167867791379e+24 kg
Parsec: 3.085677581491367e+16 m
```

As you can see, all of them has the `relevant SI units` along with the values. How is this happening? Can we also do that?

Astropy handles units and unit conversions elegantly. We will explore that in the code below.

Example 1

In [24]:
```
from astropy import units as u

# Creating quantities with units
distance = 15.2 * u.m
time = 32.4 * u.s
speed = distance / time
print(f"Speed: {speed}")

# Unit conversions
print(f"15.2 m in parsecs: {(distance).to(u.pc)}")
print(f"32.4 s in hours: {(time).to(u.hour)}")
```

```
Speed: 0.4691358024691358 m / s
15.2 m in parsecs: 4.925984519955434e-16 pc
32.4 s in hours: 0.009 h
```

Example 2

In [27]:
```python
# Composite units
energy = 500 * u.kg * u.m**2 / u.s**2
print(f"Energy in Joules: {energy}")
print(f"Energy in erg: {energy.to(u.erg)}") #multiply by 10 pow 7
```

```
Energy in Joules: 500.0 m2 kg / s2
Energy in erg: 5000000000.0 erg
```

Example 3

In [30]:
```python
# 1 light-year to various units
ly = 1 * u.lyr

print("\n1 Light-year in other units:")
print(f"Parsecs: {ly.to(u.pc):.4f}")
print(f"Meters: {ly.to(u.m):.2e}")
print(f"Astronomical Units: {ly.to(u.au):.2e}")
print(f"Kilometers: {ly.to(u.km):.2e}")
```

```
1 Light-year in other units:
Parsecs: 0.3066 pc
Meters: 9.46e+15 m
Astronomical Units: 6.32e+04 AU
Kilometers: 9.46e+12 km
```

Coordinate System

In [32]:
```python
from astropy.coordinates import SkyCoord

# Creating a coordinate from degrees, ra = right accersion 0 to 360 degree along x axis, dec = decleration -90
c1 = SkyCoord(ra=10.625 * u.degree, dec=41.2 * u.degree, frame='icrs')
print(c1)

# Creating from HMS/DMS strings
c2 = SkyCoord('00h42m30s', '+41d12m00s', frame='icrs')
print(c2)

# Galactic coordinates
c_gal = SkyCoord(l=121.1743 * u.degree, b=-21.5733 * u.degree, frame='galactic')
print(f"Galactic coordinates: {c_gal}")
print(f"Converted to ICRS: {c_gal.icrs}")
```

```
<SkyCoord (ICRS): (ra, dec) in deg
    (10.625, 41.2)>
<SkyCoord (ICRS): (ra, dec) in deg
    (10.625, 41.2)>
Galactic coordinates: <SkyCoord (Galactic): (l, b) in deg
    (121.1743, -21.5733)>
Converted to ICRS: <SkyCoord (ICRS): (ra, dec) in deg
    (10.68467197, 41.26875781)>
```

In [34]:
```python
# Calculating distance between two coordinates

# Define two points on the sky
star1 = SkyCoord('00h42m44.3s', '+41d16m09s', frame='icrs')
star2 = SkyCoord('02h42m40s', '+12d16m08s', frame='icrs')

# Calculate separation
sep = star1.separation(star2)
print(f"Separation: {sep}")
print(f"Separation in arcseconds: {sep.arcsecond}\"")
```

```
Separation: 39.07385548040611 deg
Separation in arcseconds: 140665.879729462"
```

Opening FITS files and loading the image data

FITS : Flexible Image Transport System

You can save both image and header (heading/key info) or data table in this format.

Astronomers use this format to save and analyse data.

Display FITS file

Have you seen the Horsehead Nebula?

In [40]:
```python
from astropy.visualization import astropy_mpl_style
```

```python
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
# Important
from astropy.io import fits
from astropy.utils.data import download_file
```

In [45]:
```python
# %matplotlib inline
plt.style.use(astropy_mpl_style)
image_file = download_file('http://www.astropy.org/astropy-data/tutorials/FITS-images/HorseHead.fits', cache=Tru
```

Let's open the FITS file to find out what it contains.

In [48]:
```python
hdu_list = fits.open(image_file) # hdu=  header data unit
hdu_list.info()
```

```
Filename: /Users/kritishnagyal/.astropy/cache/download/url/217b4fe80e6f349ef703ceed7e0be888/contents
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     161   (891, 893)   int16
  1  er.mask       1 TableHDU        25   1600R x 4C   [F6.2, F6.2, F6.2, F6.2]
```

Generally, the image information is located in the `PRIMARY` block. The blocks are numbered and can be accessed by indexing `hdu_list`.

In [50]:
```python
hdu_list[0].header
```

Out[50]:
```
SIMPLE  =                    T /FITS: Compliance
BITPIX  =                   16 /FITS: I*2 Data
NAXIS   =                    2 /FITS: 2-D Image Data
NAXIS1  =                  891 /FITS: X Dimension
NAXIS2  =                  893 /FITS: Y Dimension
EXTEND  =                    T /FITS: File can contain extensions
DATE    = '2014-01-09        '  /FITS: Creation Date
ORIGIN  = 'STScI/MAST'         /GSSS: STScI Digitized Sky Survey
SURVEY  = 'SERC-ER '           /GSSS: Sky Survey
REGION  = 'ER768   '           /GSSS: Region Name
PLATEID = 'A0JP    '           /GSSS: Plate ID
SCANNUM = '01      '           /GSSS: Scan Number
DSCNDNUM= '00      '           /GSSS: Descendant Number
TELESCID=                    4 /GSSS: Telescope ID
BANDPASS=                   36 /GSSS: Bandpass Code
COPYRGHT= 'AAO/ROE '           /GSSS: Copyright Holder
SITELAT =              -31.277 /Observatory: Latitude
SITELONG=              210.934 /Observatory: Longitude
TELESCOP= 'UK Schmidt - Doubl' /Observatory: Telescope
INSTRUME= 'Photographic Plate' /Detector: Photographic Plate
EMULSION= 'IIIaF   '           /Detector: Emulsion
FILTER  = 'OG590   '           /Detector: Filter
PLTSCALE=                67.20 /Detector: Plate Scale arcsec per mm
PLTSIZEX=              355.000 /Detector: Plate X Dimension mm
PLTSIZEY=              355.000 /Detector: Plate Y Dimension mm
PLATERA =        85.5994550000 /Observation: Field centre RA degrees
PLATEDEC=       -4.94660910000 /Observation: Field centre Dec degrees
PLTLABEL= 'OR14052 '           /Observation: Plate Label
DATE-OBS= '1990-12-22T13:49:00' /Observation: Date/Time
EXPOSURE=                 65.0 /Observation: Exposure Minutes
PLTGRADE= 'AD2     '           /Observation: Plate Grade
OBSHA   =             0.158333 /Observation: Hour Angle
OBSZD   =              26.3715 /Observation: Zenith Distance
AIRMASS =              1.11587 /Observation: Airmass
REFBETA =        66.3196420000 /Observation: Refraction Coeff
REFBETAP=     -0.0820000000000 /Observation: Refraction Coeff
REFK1   =        6423.52290000 /Observation: Refraction Coeff
REFK2   =       -102122.550000 /Observation: Refraction Coeff
CNPIX1  =                12237 /Scan: X Corner
CNPIX2  =                19965 /Scan: Y Corner
XPIXELS =                23040 /Scan: X Dimension
YPIXELS =                23040 /Scan: Y Dimension
XPIXELSZ=              15.0295 /Scan: Pixel Size microns
YPIXELSZ=              15.0000 /Scan: Pixel Size microns
PPO1    =       -3069417.00000 /Scan: Orientation Coeff
PPO2    =       0.000000000000 /Scan: Orientation Coeff
PPO3    =        177500.000000 /Scan: Orientation Coeff
PPO4    =       0.000000000000 /Scan: Orientation Coeff
PPO5    =        3069417.00000 /Scan: Orientation Coeff
PPO6    =        177500.000000 /Scan: Orientation Coeff
PLTRAH  =                    5 /Astrometry: Plate Centre H
PLTRAM  =                   42 /Astrometry: Plate Centre M
PLTRAS  =                23.86 /Astrometry: Plate Centre S
PLTDECSN= '-       '           /Astrometry: Plate Centre +/-
PLTDECD =                    4 /Astrometry: Plate Centre D
PLTDECM =                   56 /Astrometry: Plate Centre M
```

```
PLTDECS =                 47.9 /Astrometry: Plate Centre S
EQUINOX =               2000.0 /Astrometry: Equinox
AMDX1   =        67.1550859799 /Astrometry: GSC1 Coeff
AMDX2   =      0.0431478884485 /Astrometry: GSC1 Coeff
AMDX3   =       -292.435619180 /Astrometry: GSC1 Coeff
AMDX4   =  -2.68934864702E-005 /Astrometry: GSC1 Coeff
AMDX5   =   1.99133423290E-005 /Astrometry: GSC1 Coeff
AMDX6   =  -2.37011931379E-006 /Astrometry: GSC1 Coeff
AMDX7   =       0.000000000000 /Astrometry: GSC1 Coeff
AMDX8   =   2.21426387429E-006 /Astrometry: GSC1 Coeff
AMDX9   =  -8.12841581455E-008 /Astrometry: GSC1 Coeff
AMDX10  =   2.48169090021E-006 /Astrometry: GSC1 Coeff
AMDX11  =   2.77618933926E-008 /Astrometry: GSC1 Coeff
AMDX12  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDX13  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDX14  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDX15  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDX16  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDX17  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDX18  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDX19  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDX20  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDY1   =        67.1593591466 /Astrometry: GSC1 Coeff
AMDY2   =     -0.0471363749174 /Astrometry: GSC1 Coeff
AMDY3   =        316.004963520 /Astrometry: GSC1 Coeff
AMDY4   =   2.86798151430E-005 /Astrometry: GSC1 Coeff
AMDY5   =  -2.00968236347E-005 /Astrometry: GSC1 Coeff
AMDY6   =   2.27840393227E-005 /Astrometry: GSC1 Coeff
AMDY7   =       0.000000000000 /Astrometry: GSC1 Coeff
AMDY8   =   2.23885090381E-006 /Astrometry: GSC1 Coeff
AMDY9   =  -2.28360163464E-008 /Astrometry: GSC1 Coeff
AMDY10  =   2.44828851495E-006 /Astrometry: GSC1 Coeff
AMDY11  =  -5.76717487998E-008 /Astrometry: GSC1 Coeff
AMDY12  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDY13  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDY14  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDY15  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDY16  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDY17  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDY18  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDY19  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDY20  =       0.000000000000 /Astrometry: GSC1 Coeff
AMDREX1 =        67.1532034737 /Astrometry: GSC2 Coeff
AMDREX2 =      0.0434354199559 /Astrometry: GSC2 Coeff
AMDREX3 =       -292.435438892 /Astrometry: GSC2 Coeff
AMDREX4 =   4.60919247070E-006 /Astrometry: GSC2 Coeff
AMDREX5 =  -3.21138058537E-006 /Astrometry: GSC2 Coeff
AMDREX6 =   7.23651736725E-006 /Astrometry: GSC2 Coeff
AMDREX7 =       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX8 =       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX9 =       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX10=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX11=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX12=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX13=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX14=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX15=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX16=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX17=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX18=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX19=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREX20=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY1 =        67.1522589487 /Astrometry: GSC2 Coeff
AMDREY2 =     -0.0481758265285 /Astrometry: GSC2 Coeff
AMDREY3 =        315.995683716 /Astrometry: GSC2 Coeff
AMDREY4 =  -7.47397531230E-006 /Astrometry: GSC2 Coeff
AMDREY5 =   9.55221105409E-007 /Astrometry: GSC2 Coeff
AMDREY6 =   7.60954485251E-006 /Astrometry: GSC2 Coeff
AMDREY7 =       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY8 =       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY9 =       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY10=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY11=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY12=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY13=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY14=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY15=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY16=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY17=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY18=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY19=       0.000000000000 /Astrometry: GSC2 Coeff
AMDREY20=       0.000000000000 /Astrometry: GSC2 Coeff
ASTRMASK= 'er.mask '           /Astrometry: GSC2 Mask
```

```
WCSAXES =                    2 /GetImage: Number WCS axes
WCSNAME = 'DSS               ' /GetImage: Local WCS approximation from full plat
RADESYS = 'ICRS              ' /GetImage: GSC-II calibration using ICRS system
CTYPE1  = 'RA---TAN          ' /GetImage: RA-Gnomic projection
CRPIX1  =           446.000000 /GetImage: X reference pixel
CRVAL1  =            85.274970 /GetImage: RA of reference pixel
CUNIT1  = 'deg               ' /GetImage: degrees
CTYPE2  = 'DEC--TAN          ' /GetImage: Dec-Gnomic projection
CRPIX2  =           447.000000 /GetImage: Y reference pixel
CRVAL2  =            -2.458265 /GetImage: Dec of reference pixel
CUNIT2  = 'deg               ' /Getimage: degrees
CD1_1   =        -0.0002802651 /GetImage: rotation matrix coefficient
CD1_2   =         0.0000003159 /GetImage: rotation matrix coefficient
CD2_1   =         0.0000002767 /GetImage: rotation matrix coefficient
CD2_2   =         0.0002798187 /GetImage: rotation matrix coefficient
OBJECT  = 'data              ' /GetImage: Requested Object Name
DATAMIN =                 3759 /GetImage: Minimum returned pixel value
DATAMAX =                22918 /GetImage: Maximum returned pixel value
OBJCTRA = '05 41 06.000      ' /GetImage: Requested Right Ascension (J2000)
OBJCTDEC= '-02 27 30.00      ' /GetImage: Requested Declination (J2000)
OBJCTX  =             12682.48 /GetImage: Requested X on plate (pixels)
OBJCTY  =             20411.37 /GetImage: Requested Y on plate (pixels)
```

.

In [52]:
```
image_data = hdu_list[0].data
image_data
```

Out[52]:
```
array([[ 7201,  6642,  6642, ...,  9498,  9498, 10057],
       [ 6642,  6363,  6642, ..., 10057, 10616, 10616],
       [ 6922,  6642,  6922, ..., 10337, 11175, 10616],
       ...,
       [ 5412,  5132,  5412, ..., 13000, 12580, 12021],
       [ 5796,  5517,  5796, ..., 12546, 12546, 11987],
       [ 5796,  5796,  6076, ..., 11987, 12546, 12546]], dtype='>i2')
```

Our data is now stored as a 2D numpy array. But how do we know the dimensions of the image? We can look at the `shape` of the array.

In [56]:
```
print(type(image_data))
print(image_data.shape) # y,x rather x,y
```

```
<class 'numpy.ndarray'>
(893, 891)
```

In [58]:
```
image_data1 = fits.getdata(image_file)
image_data1
# print(image_data.shape)
```

Out[58]:
```
array([[ 7201,  6642,  6642, ...,  9498,  9498, 10057],
       [ 6642,  6363,  6642, ..., 10057, 10616, 10616],
       [ 6922,  6642,  6922, ..., 10337, 11175, 10616],
       ...,
       [ 5412,  5132,  5412, ..., 13000, 12580, 12021],
       [ 5796,  5517,  5796, ..., 12546, 12546, 11987],
       [ 5796,  5796,  6076, ..., 11987, 12546, 12546]], dtype='>i2')
```

In [60]:
```
np.array_equal(image_data,image_data1)
```

Out[60]:
```
True
```

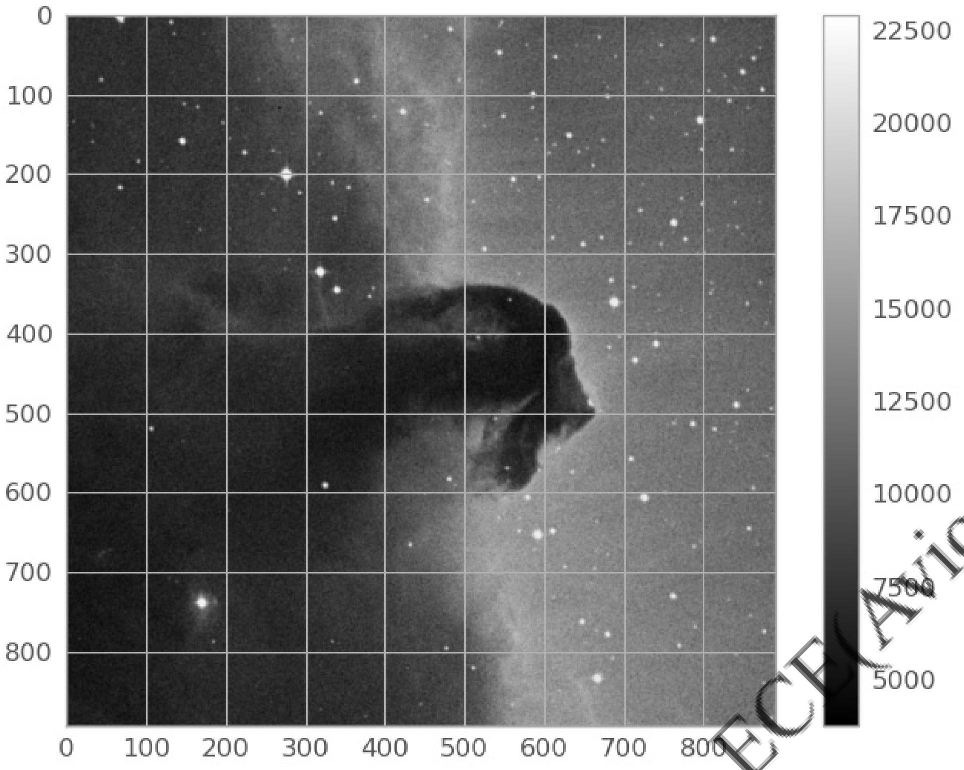
Great! At this point, we can close the FITS file because we've stored everything we wanted to a variable.

In [63]:
```
hdu_list.close()
```

Viewing the image data and getting basic statistics

In [70]:
```
# %matplotlib inline
plt.imshow(image_data, cmap='gray')  #cmap means colormap
plt.colorbar()
```

Out[70]:
```
<matplotlib.colorbar.Colorbar at 0x16c56e250>
```



For more color maps http://wiki.scipy.org/Cookbook/Matplotlib/Show_colormaps

Let's get some basic statistics about our image:

In [82]:
```
print('Min:', np.min(image_data))
print('Max:', np.max(image_data))
print('Mean:', np.mean(image_data))
print('Std deviation:', np.std(image_data))
```

```
Min: 3759
Max: 22918
Mean: 9831.481676287574
Std deviation: 3032.3927542049046
```

Plotting a histogram

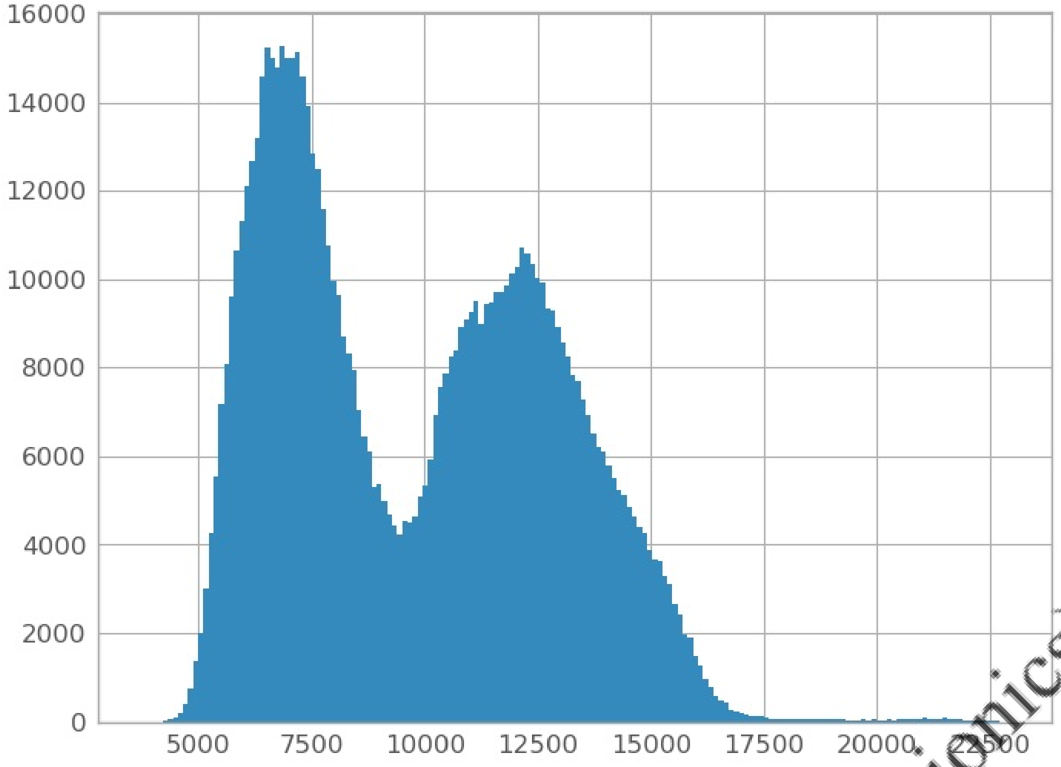
To make a histogram with `matplotlib.pyplot.hist()`, we'll need to cast the data from a 2D array to something one dimensional.

In this case, let's use the `ndarray.flatten()` to return a 1D numpy array.

In [76]:
```
image_data.flatten()
```

Out[76]:
```
array([ 7201,  6642,  6642, ..., 11987, 12546, 12546], dtype='>i2')
```

In [78]:
```
histogram = plt.hist(image_data.flatten(), bins='auto')
```



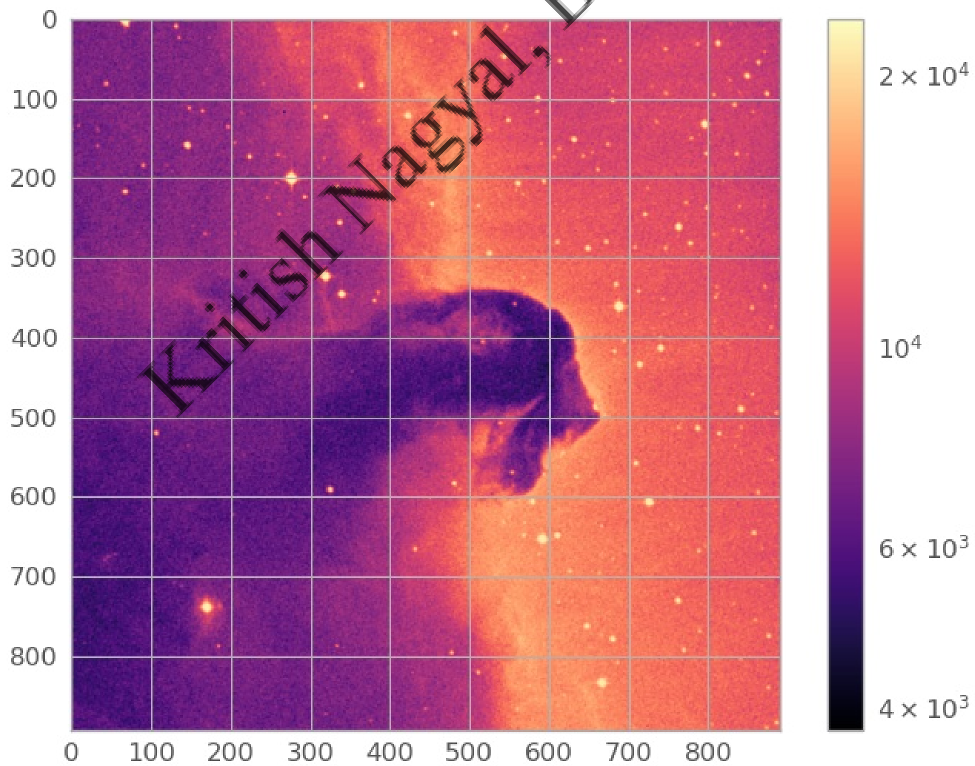
Same thing but logarithmic!

What if we want to use a logarithmic color scale? To do so, we'll need to load the `LogNorm` object from `matplotlib`.

In [86]:
```
from matplotlib.colors import LogNorm
```

In [88]:
```
plt.imshow(image_data, cmap='magma', norm=LogNorm())
plt.colorbar()
```

Out[88]:
```
<matplotlib.colorbar.Colorbar at 0x16c85e250>
```



Do you know why this is better? Because color makes it more visually clear and `log scale` enhances even the `small variation`.

For example: 1e-5 and 1e-4, although small but its log is -5 and -4 (Quite distinct now !!).

M42 - Orion Nebula

https://dr12.sdss.org/fields

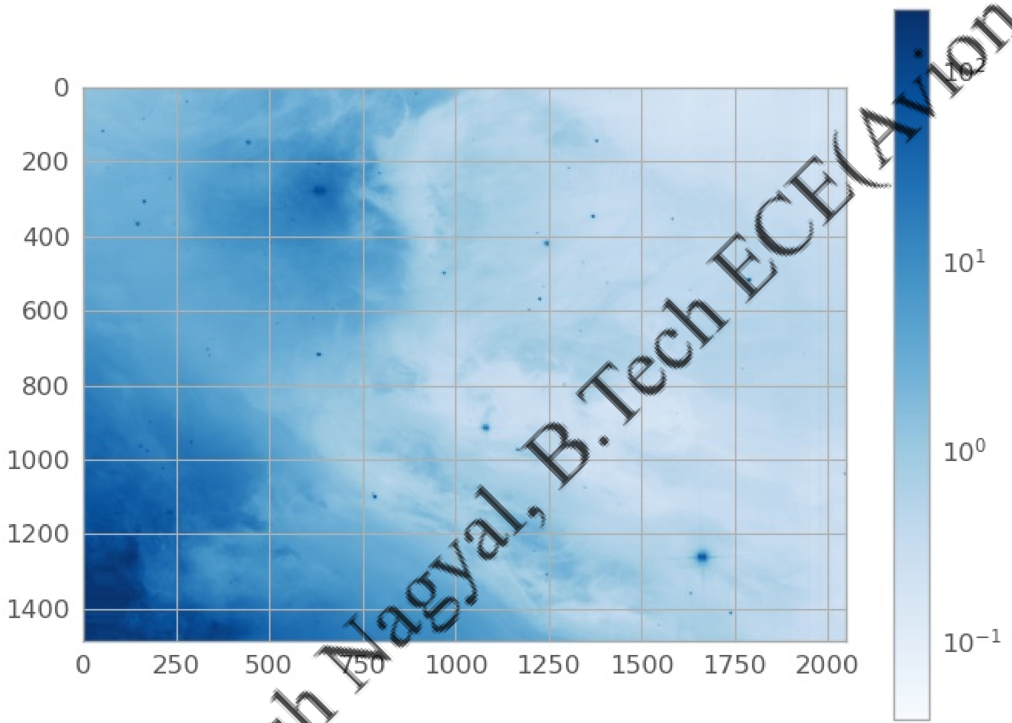
In [97]:
```python
g=fits.open("frame-g-006073-4-0063.fits")
g[0].header
```

Out[97]:
```
SIMPLE  =                    T /
BITPIX  =                  -32 / 32 bit floating point
NAXIS   =                    2
NAXIS1  =                 2048
NAXIS2  =                 1489
EXTEND  =                    T /Extensions may be present
BZERO   =              0.00000 /Set by MRD_SCALE
BSCALE  =              1.00000 /Set by MRD_SCALE
TAI     =        4649973400.67 / 1st row Number of seconds since Nov 17 1858
RA      =            83.820000 / 1st row RA of telescope boresight (deg)
DEC     =            -5.947992  / 1st row Dec of telescope boresight (degrees)
SPA     =             180.000  / 1st row Cam col position angle wrt N (deg)
IPA     =             137.106  / 1st row Inst rotator position angle (deg)
IPARATE =             -0.0006  / 1st row Inst rotator anglr velocity (deg/sec)
AZ      =            306.29262 / 1st row Azimuth (encoder) of tele (0=N?) (deg)
ALT     =            34.529078 / 1st row Altitude (encoder) of tele (degrees)
FOCUS   =           -169.90000 / 1st row - Focus piston (microns?)
DATE-OBS= '2006-03-25'         / 1st row - TAI date
TAIHMS  = '03:16:40.66'        / 1st row TAI time HH:MM:SS.SS
COMMENT  TAI,RA,DEC,SPA,IPA,IPARATE,AZ,ALT,FOCUS at reading of col 0, row 0
ORIGIN  = 'SDSS    '
TELESCOP= '2.5m    '
TIMESYS = 'TAI     '
RUN     =                 6073 / Run number
FRAME   =                   71 / Frame sequence number within the run
CCDLOC  =                   54 / Survey location of CCD (e.g., rowCol)
STRIPE  =                  213 / Stripe index number (23 <--> eta=0)
STRIP   = 'S       '           / Strip in the stripe being tracked.
FLAVOR  = 'calibration'        / Flavor of this run
OBSERVER= 'viktorm '           / Observer
SYS_SCN = 'mean    '           / System of the scan great circle (e.g., mean)
EQNX_SCN=            2000.00   / Equinox of the scan great circle. (years)
NODE    =            0.00000   / RA of the great circle's ascending node (deg)
INCL    =            0.00000   / Great circle's inclination wrt cel. eq. (deg)
XBORE   =            22.74     / Boresight x offset from the array center (mm)
YBORE   =            0.00      / Boresight x offset from the array center (mm)
OBJECT  = '213 S   '           / e.g., 'stripe 50.6 degrees, north strip'
EXPTIME = '53.907456'          / Exposure time (seconds)
SYSTEM  = 'FK5     '           / System of the TCC coordinates (e.g., mean)
CCDMODE = 'DRIFT   '           / 'STARING' or 'DRIFT'
C_OBS   =                26322 / CCD row clock rate (usec/row)
COLBIN  =                    1 / Binning factor perpendicular to the columns
ROWBIN  =                    1 / Binning factor perpendicular to the rows
DAVERS  = 'v15_11  '           / Version of DA software
SCDMETHD= 'sqrtDynamic'        / scdMethod
SCDWIDTH=                 1280 / scdDisplayWidth
SCDDECMF=                    1 / scdDecimateFactor
SCDOFSET=                  410 / scdDisplayOffset
SCDDYNTH=                  -21 / scdDynamicThresh
SCDSTTHL=                   30 / scdStaticThreshL
SCDSTTHR=                   30 / scdStaticThreshR
SCDREDSZ=                  527 / scdReduceSize
SCDSKYL =                 1247 / scdSkyLeft
SCDSKYR =                 1214 / scdSkyRight
COMMENT  CCD-specific parameters
CAMROW  =                    5 / Row in the imaging camera
BADLINES=                    0 / Number of bad lines in frame
EQUINOX =              2000.00 /
SOFTBIAS=                 1000 / software "bias" added to all DN
BUNIT   = 'nanomaggy'          / 1 nanomaggy = 3.631e-6 Jy
FILTER  = 'g       '           / filter used
CAMCOL  =                    4 / column in the imaging camera
VERSION = 'v5_6_3  '
DERV_VER= 'NOCVS:v8_23'
ASTR_VER= 'NOCVS:v5_24'
ASTRO_ID= '2010-03-29T20:33:41 25639'
BIAS_ID = 'PS      '
FRAME_ID= '2010-08-10T09:06:37 03057'
KO_VER  = 'devel   '
PS_ID   = '2010-03-29T20:12:50 25067 camCol 4'
ATVSN   = 'NOCVS:v5_24'        / ASTROTOOLS version tag
RADECSYS= 'ICRS    '           / International Celestial Ref. System
CTYPE1  = 'RA---TAN'           /Coordinate type
CTYPE2  = 'DEC--TAN'           /Coordinate type
CUNIT1  = 'deg     '           /Units
CUNIT2  = 'deg     '           /Units
CRPIX1  =        1025.00000000 /X of reference pixel
CRPIX2  =        745.000000000 /Y of reference pixel
```

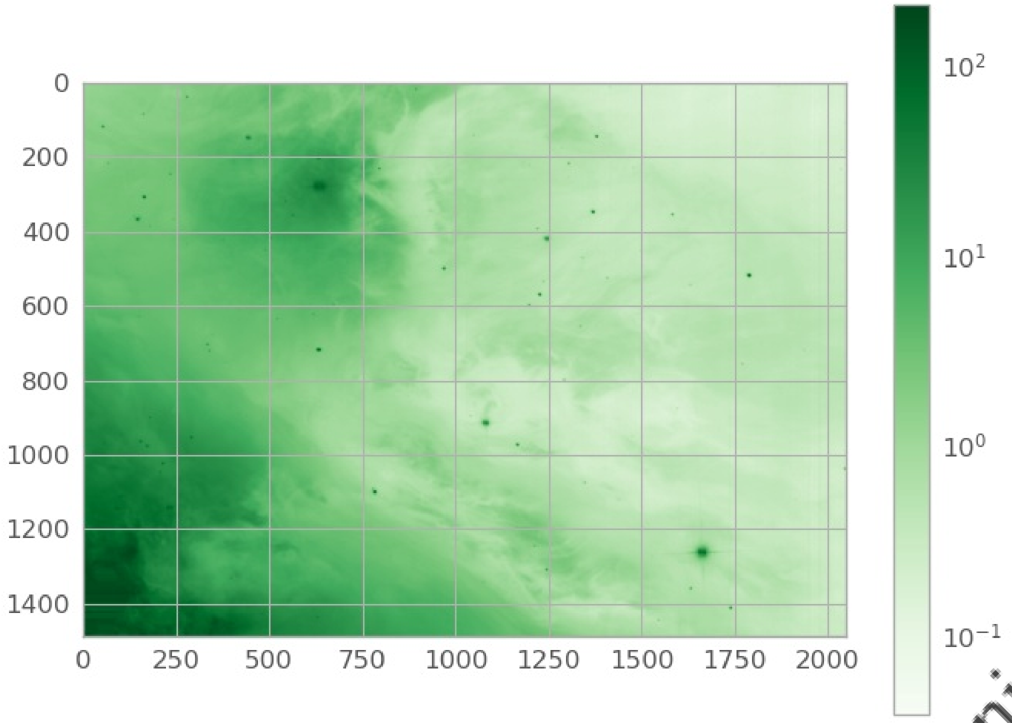
```
CRVAL1  =        83.9237640908 /RA of reference pixel (deg)
CRVAL2  =       -5.31827757943 /Dec of reference pixel (deg)
CD1_1   =    0.000109925385690 /RA deg per column pixel
CD1_2   =    2.27483771892E-08 /RA deg per row pixel
CD2_1   =    1.99364884357E-07 /Dec deg per column pixel
CD2_2   =   -0.000109997185155 /Dec deg per row pixel
HISTORY GSSSPUTAST: Aug 10 09:06:46 2010
COMMENT  Calibration parameters
COMMENT  Floats truncated at 10 binary digits with FLOATCOMPRESS
NMGY    =           0.00408423 / Calibration factor [nMgy per count]
NMGYIVAR=            0.0709756 / Calibration factor inverse variance
VERSIDL = '7.0     '           / Version of IDL
VERSUTIL= 'v5_5_5  '           / Version of idlutils
VERSPOP = 'v1_11_1 '           / Version of photoop product
PCALIB  = '/clusterfs/riemann/raid006/dr8/groups/boss/calib/dr8_final' / Value o
PSKY    = '/clusterfs/riemann/raid006/dr8/groups/boss/photo/sky' / Value of PHOT
RERUN   = '301     '           / rerun
HISTORY SDSS_FRAME_ASTROM: Astrometry fixed for dr9 Sun Jun 24 23:13:10 2012
```

In [99]:
```
# plt.figure()
# plt.imshow(fits.getdata('frame-g-005183-2-0406.fits'), cmap='Greens')
# image_hist = plt.hist(fits.getdata('frame-g-005183-2-0406.fits').flatten(), bins='auto')
```

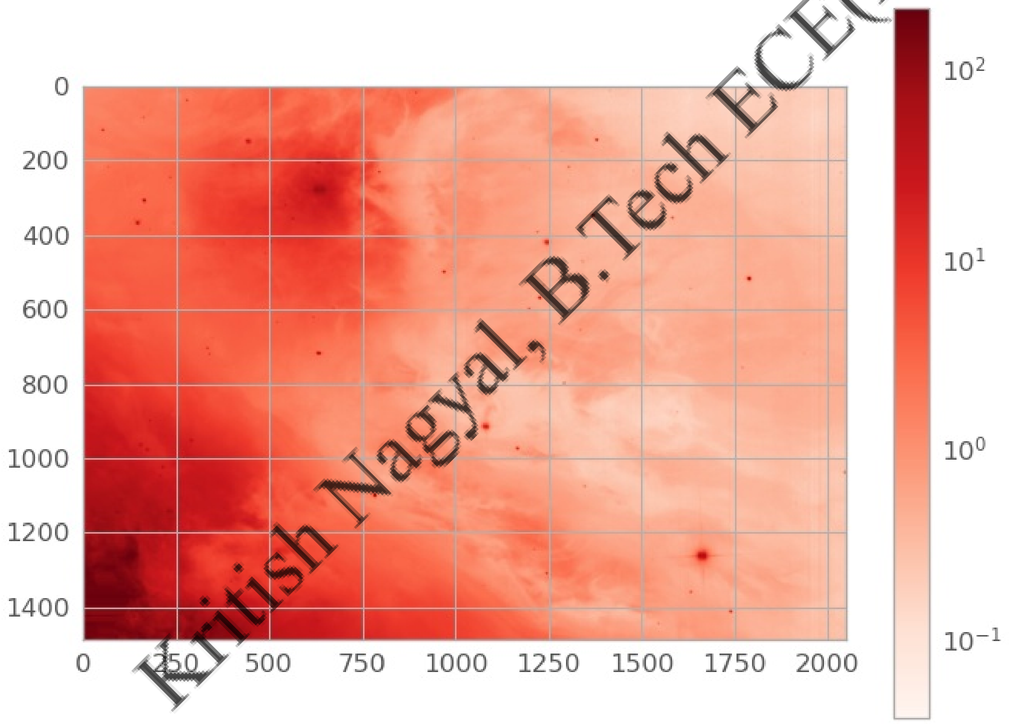
In [103…
```
plt.figure()
plt.imshow(fits.getdata('frame-g-006073-4-0063.fits'),norm=LogNorm(),cmap="Blues")
cbar = plt.colorbar()
```



In [107…
```
plt.figure()
plt.imshow(fits.getdata('frame-g-006073-4-0063.fits'),norm=LogNorm(),cmap="Greens")
cbar = plt.colorbar()
```



In [111…
```
plt.figure()
plt.imshow(fits.getdata("frame-g-006073-4-0063.fits"),norm=LogNorm(),cmap="Reds")
cbar = plt.colorbar()
```

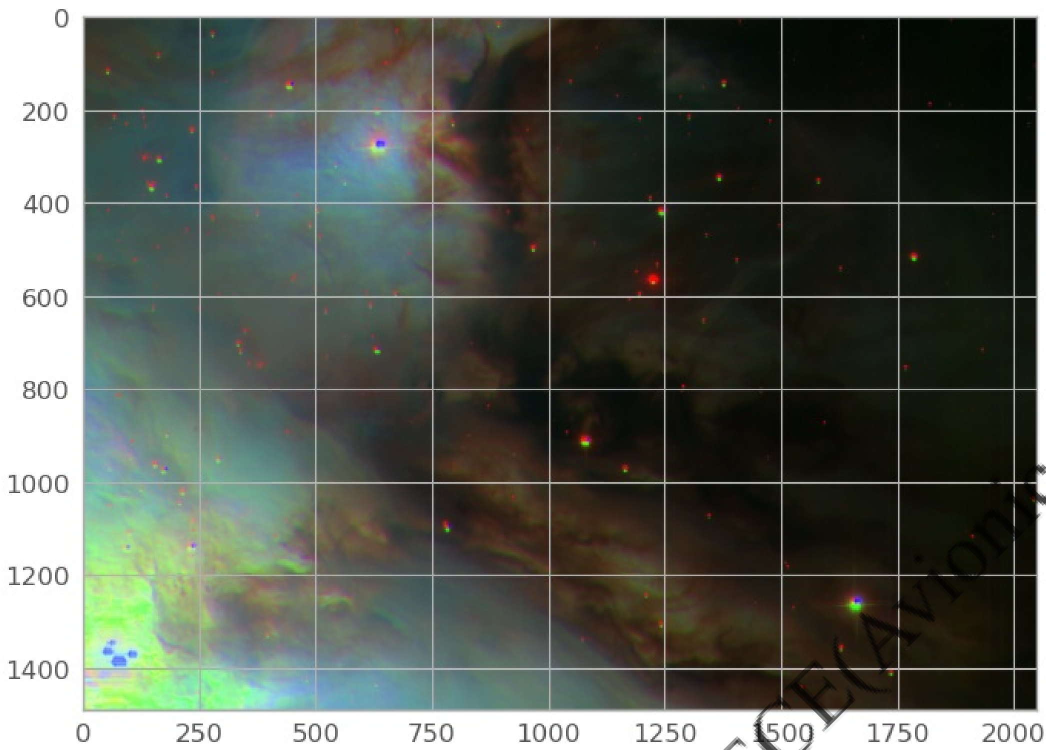


Combining into colourful from individual filters

In [115…
```
from astropy.visualization import make_lupton_rgb
```

In [121…
```
b=fits.getdata('frame-u-006073-4-0063.fits')
r=fits.getdata('frame-i-006073-4-0063.fits')
g=fits.getdata('frame-g-006073-4-0063.fits')
plt.figure()
plt.imshow(make_lupton_rgb(r,g,b))
```

Out[121… <matplotlib.image.AxesImage at 0x16ddbb690>



Convert jpg to fits

Load and display the original 3-color jpeg image:

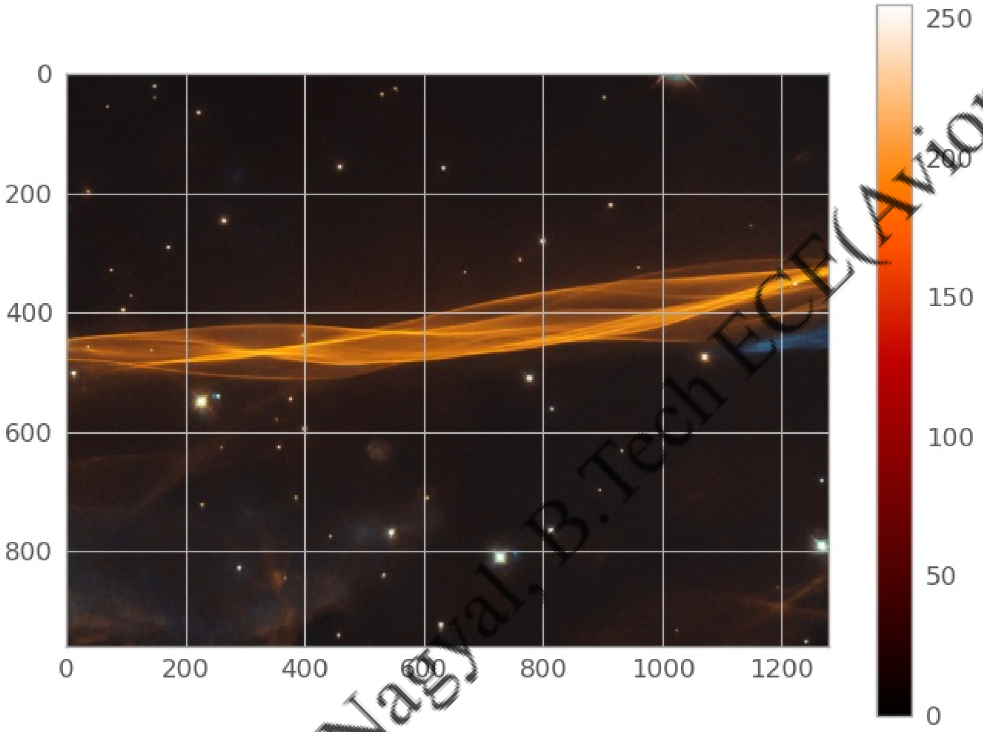
In [127…

```
# %matplotlib inline
import numpy as np
from PIL import Image
from astropy.io import fits
import matplotlib.pyplot as plt
# from astropy.visualization import astropy_mpl_style
# plt.style.use(astropy_mpl_style)
image = Image.open('hubble_image.jpeg')
xsize, ysize = image.size
print("Image size: {} x {}".format(xsize, ysize))
plt.imshow(image)
np.array(image.getdata())
plt.colorbar()
```

```
Image size: 1280 x 960
```

Out[127…

```
<matplotlib.colorbar.Colorbar at 0x16f115590>
```



In [129…

```
r, g, b = image.split()
r_data = np.array(r.getdata()) # data is now an array of length ysize*xsize
g_data = np.array(g.getdata())
b_data = np.array(b.getdata())
print(r_data.shape)
r_data
```

```
(1228800,)
```

Out[129…

```
array([25, 25, 24, ..., 27, 27, 28])
```

Split the three channels (RGB) and get the data as Numpy arrays. The arrays are flattened, so they are 1-dimensional:

In [138…

```
r_data = r_data.reshape(ysize, xsize)
g_data = g_data.reshape(ysize, xsize)
b_data = b_data.reshape(ysize, xsize)

red = fits.PrimaryHDU(data=r_data)
red.writeto('red.fits',overwrite=True)

green = fits.PrimaryHDU(data=g_data)
green.writeto('green.fits',overwrite=True)

blue = fits.PrimaryHDU(data=b_data)
blue.writeto('blue.fits',overwrite=True)
```

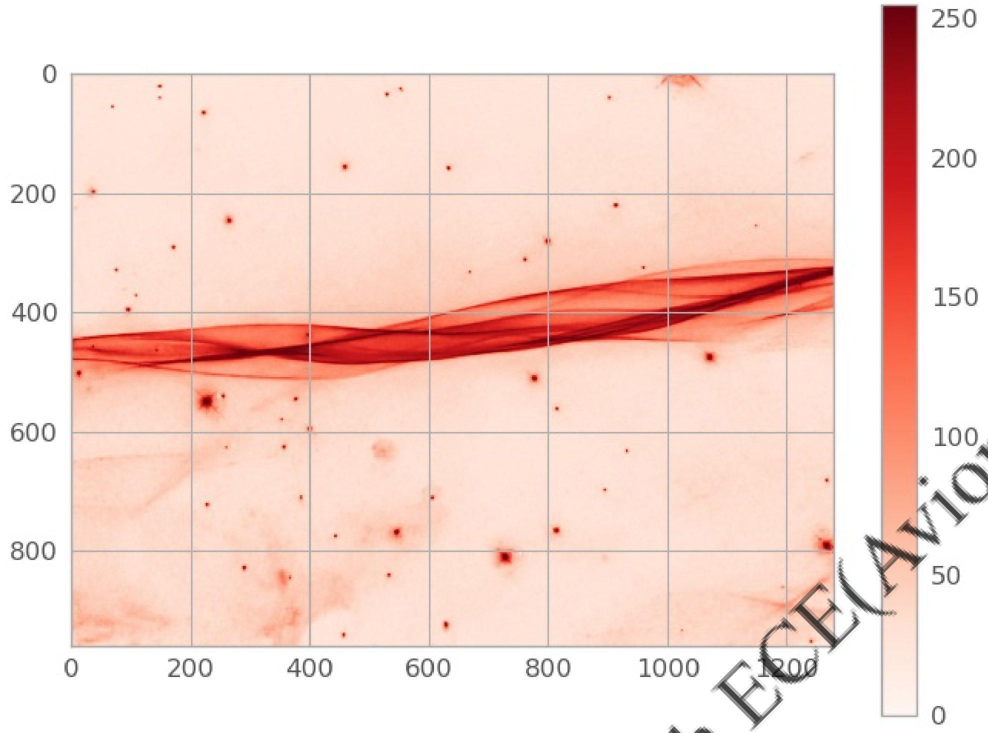
Display are three

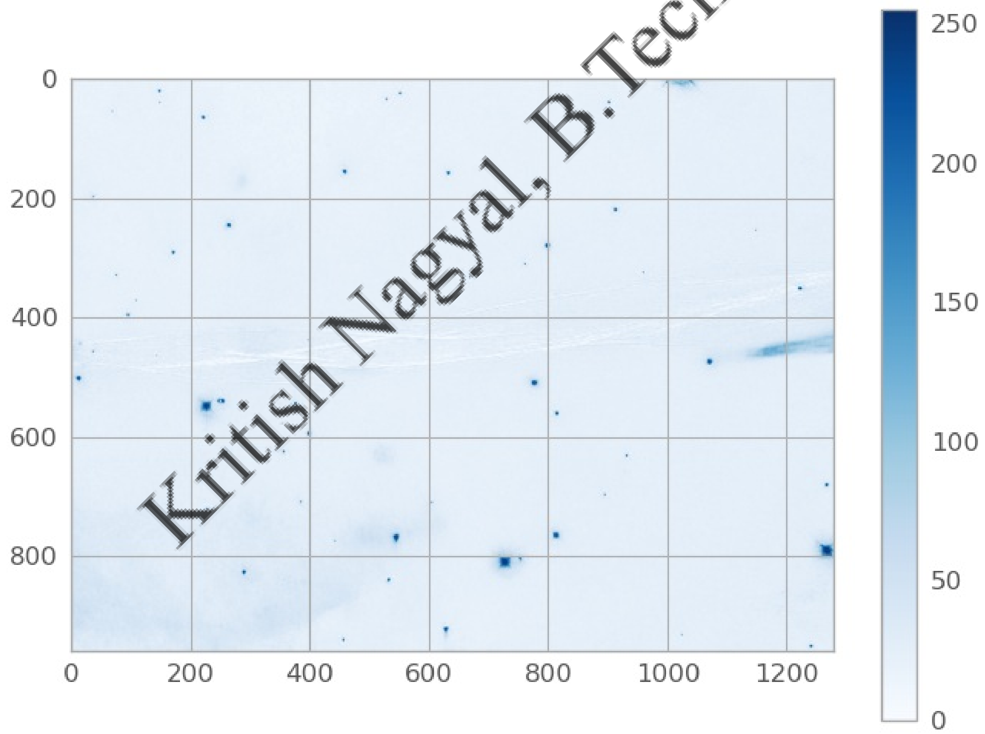
In [134…

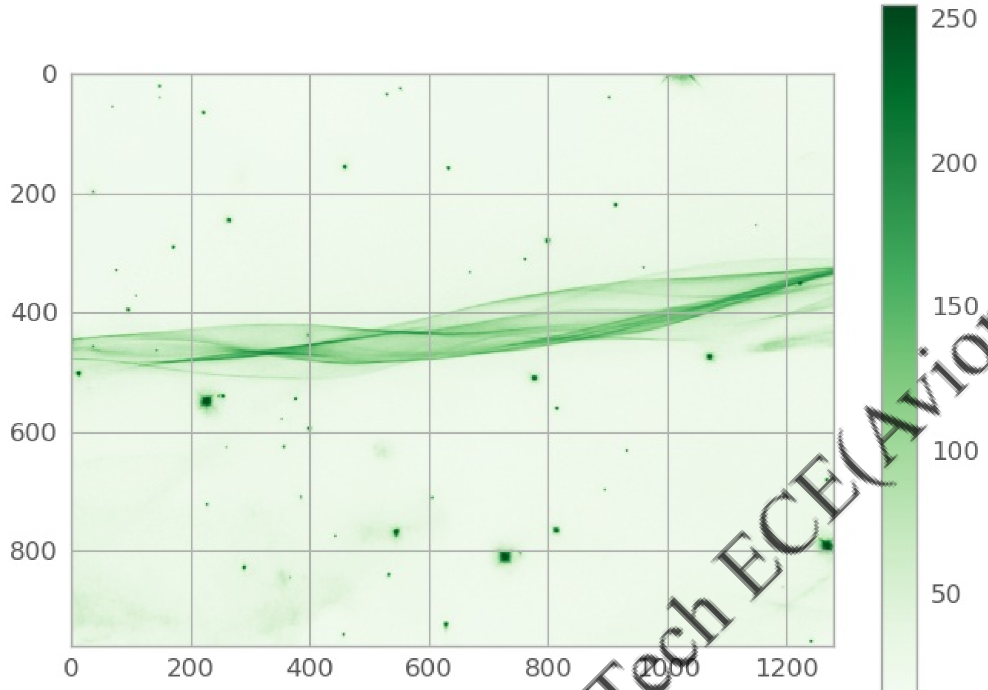
```
plt.figure()
```

```python
plt.imshow(fits.getdata('red.fits'),cmap="Reds")
plt.colorbar()
plt.figure()
plt.imshow(fits.getdata('blue.fits'),cmap="Blues")
plt.colorbar()
plt.figure()
plt.imshow(fits.getdata('green.fits'),cmap="Greens")
plt.colorbar()
```

Out[134… <matplotlib.colorbar.Colorbar at 0x175680510>







In [141…

```
red_image=fits.open('red.fits')
red_image[0].header
```

Out[141…

```
SIMPLE  =                    T / conforms to FITS standard
BITPIX  =                   64 / array data type
NAXIS   =                    2 / number of array dimensions
NAXIS1  =                 1280
NAXIS2  =                  960
EXTEND  =                    T
```

Writing into FITS

In [144…

```
hdulist = fits.open('red.fits')
# hdulist.info()
hdulist[0].header
# hdulist.close()
```

Out[144…

```
SIMPLE  =                    T / conforms to FITS standard
BITPIX  =                   64 / array data type
NAXIS   =                    2 / number of array dimensions
NAXIS1  =                 1280
NAXIS2  =                  960
EXTEND  =                    T
```

In [148…

```
fits.setval('red.fits', 'OBSERVER', value='Kritish Nagyal')
fits.setval('red.fits', 'Date', value='5-06-2025')
fits.setval('red.fits', 'AUTHOR', value='ISA')
```

In [30]:

```
hdulist = fits.open('red.fits')
# hdulist.info()
HDU=hdulist[0].header
HDU
# hdulist.close()
```

Out[30]:
```
SIMPLE  =                    T / conforms to FITS standard
BITPIX  =                   64 / array data type
NAXIS   =                    2 / number of array dimensions
NAXIS1  =                 1280
NAXIS2  =                  960
EXTEND  =                    T
OBSERVER= 'Satyapriya Das'
DATE    = '5-06-2025'
AUTHOR  = 'ISA     '
```

In [150…
```
fits.setval('red.fits', 'Message', value='Today is 5th June',comment="Fits file created by Satyapriya Das")
```

In [152…
```
hdulist = fits.open('red.fits')
# hdulist.info()
HDU=hdulist[0].header
HDU
# hdulist.close()
```

Out[152…
```
SIMPLE  =                    T / conforms to FITS standard
BITPIX  =                   64 / array data type
NAXIS   =                    2 / number of array dimensions
NAXIS1  =                 1280
NAXIS2  =                  960
EXTEND  =                    T
OBSERVER= 'Kritish Nagyal'
DATE    = '5-06-2025'
AUTHOR  = 'ISA     '
MESSAGE = 'Today is 5th June'  / Fits file created by Satyapriya Das
```

In [ ]: