lab_2_ntbk

March 11, 2022

1 ASP3231 Laboratory 2: Basics of Reducing CCD Images

1.1 Prelab questions

Read this Jupyter notebook in full and then complete the following prelab questions in this Jupyter notebook, and then submit a PDF of your (still incomplete) Jupyter notebook before the start of your lab session. For this lab the first two lectures and Section 4.5 of Rieke's "Measuring the Universe" are particularly helpful.

Prelab Question 1. How would you take a bias, dark, and flat image at the telescope?

Prelab Question 2. In words describe the mathematical operations that we undertake on a science image when process it using the bias, dark and flat images.

Prelab Question 3. A raw science image will include contributions from the bias (offset) and dark. How does the impact of the bias and dark vary with exposure time?

Prelab Question 4. If you wanted to use astropy to determine and then print the value the OBJECT keyword in the FITS header (i.e. from the FITS metadata), what command(s) would you use?

You still have another prelab question - see below

1. **Bias** take zero second exposure with closed shutters. **Dark** Take a long 600 second exposure with the shutters closed. **Flat** Take an exposure in the twilight sky, or against an illuminated screen
2. In general, we subtract the bias from the darks, then the bias and darks both from the offset. Finally, we divide the science exposures by the flats.
3. The bias and dark are contributors to observed Poisson noise. Increasing the exposure time can decrease the proportional contribution of the noise and increase the signal to noise ratio.
4. "' from astropy.io import fits

with fits.open('NGC_2997_R_00005534.fits') as hdul: print(hdul[0].header['OBJECT'])

For the next prelab question we need to import libraries

[ ]:
```
import numpy as np
import astropy
import ccdproc
from ccdproc import CCDData, combiner
```

```
from astropy import units as u
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
import gc                                    # What does this library do?
gc.enable()
```

Prelab Question 5. Last week, you learned how to read a FITS image into a Jupyter Notebook and display it. But you may have noticed that the fits image was small and there were no axis labels, title or caption. In this task, you will need to read in and display 'M51.fits' from the Lab 2 Moodle Folder. (M51 is also known as the Whirlpool galaxy). In the following code cell and markdown cell, your TA has started writing a code to display the image and provide a caption. You will need to complete the code using the skills you learned last week and the following matplotlib tutorial. To complete this task you will to make sure that you have: - loaded the FITS image - change the figure size and the font size - labeled the figure and axes - changed the figure colourscale

Several useful resources for this quesiton include: - https://www.datacamp.com/community/tutorials/matplotlib-tutorial-python - https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html - https://docs.astropy.org/en/stable/io/fits/

> The gc library provides garbage collection to ensure that the interpreter isn't allocated more memory than it needs

[ ]:
```
image_1 = CCDData.read("M51.fits", unit="adu")



fig, ax = plt.subplots(figsize = (11,10))

# Look through here to get some funky colour maps:
# https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html
plt.rcParams.update({'font.size': 10})
plt.imshow(image_1, cmap='gray', norm=LogNorm())
plt.xlabel('x coordinate (pixels)')
plt.ylabel('y coordinate (pixels)')
plt.title('2003-03-06 54 second exposure of M51 with 2.5m telescope')
plt.colorbar()
```

```
INFO: using the unit adu passed to the FITS reader instead of the unit nanomaggy
in the FITS file. [astropy.nddata.ccddata]
```

[ ]:
```
<matplotlib.colorbar.Colorbar at 0x28fee1db0>
```

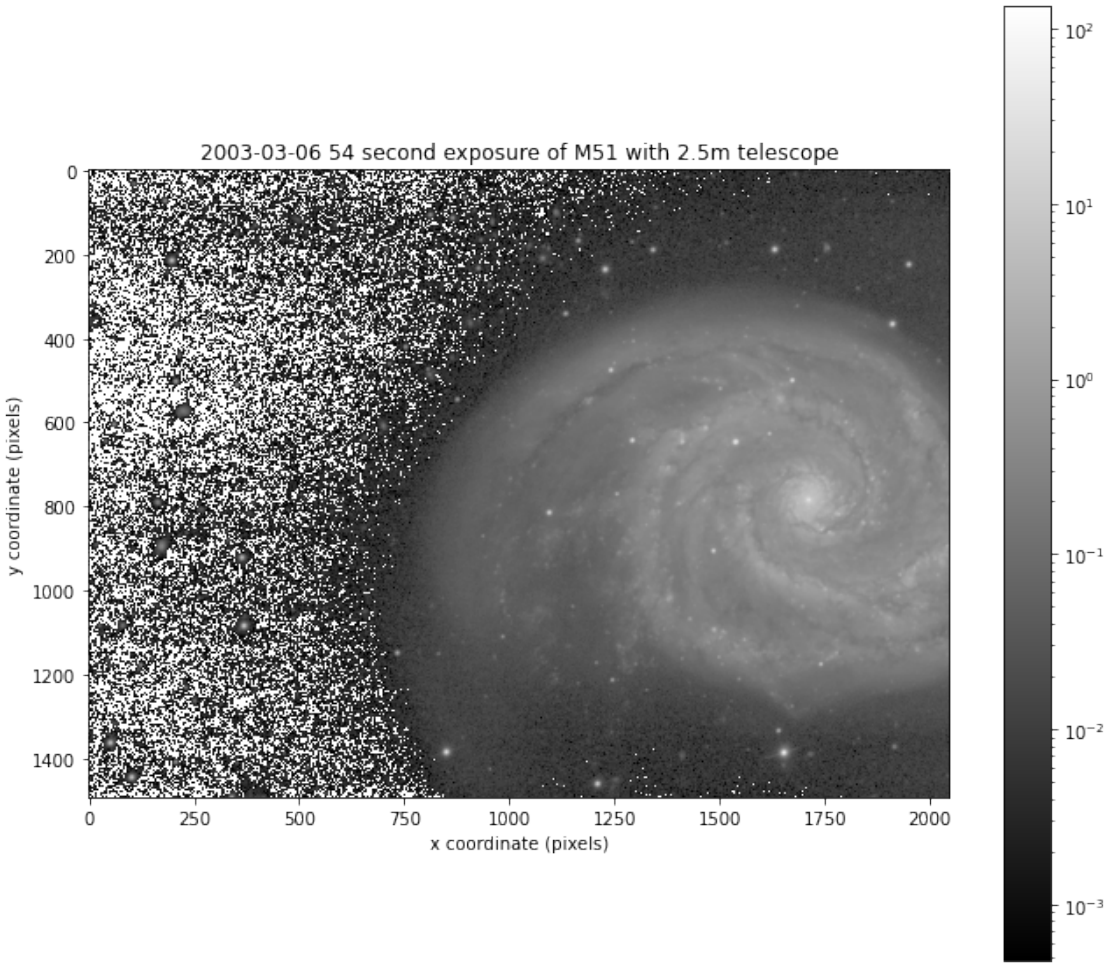


****Figure 1**** This should be a figure caption but at the moment it is placeholder text.

1.2 End of Pre-Lab

2 Lab 2. Basics of Reducing CCD Images

2.0.1 In this laboratory we will undertake the basics of data reduction (processing) of CCD images. Before the lab remind yourself what these steps are.

By the end of this lab, you should be able to: - take raw CCD images from a telescope and process them to produce processed science images - be able to describe why each step is required to produce processed science images

Task 1. Most labs use many images, so it makes sense to create image lists rather than loading images individually. Let's do this now. Run the following code cell. (Note that while this lab contains many tasks, most take just a minute or two although some do take longer.)

```
[ ]: images = ccdproc.ImageFileCollection(".")
     print(dir(images))
     print(images.files)
```

```
['__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__', '_all_keywords',
'_dict_from_fits_header', '_ext', '_filenames', '_files',
'_find_fits_by_reading', '_find_keywords_by_values', '_fits_files_in_directory',
'_fits_summary', '_generator', '_get_files', '_glob_exclude', '_glob_include',
'_location', '_paths', '_set_column_name_case_to_match_keywords', '_summary',
'ccds', 'data', 'ext', 'files', 'files_filtered', 'filter', 'glob_exclude',
'glob_include', 'hdus', 'headers', 'keywords', 'location', 'refresh', 'sort',
'summary', 'values']
['Bias_00005290.fits', 'Bias_00005291.fits', 'Bias_00005292.fits',
'Bias_00005293.fits', 'Bias_00005294.fits', 'Bias_00005295.fits',
'Bias_00005296.fits', 'Bias_00005297.fits', 'Bias_00005298.fits',
'Bias_00005299.fits', 'Dark_3600.000secs00002950.fits',
'Dark_3600.000secs00002951.fits', 'Dark_3600.000secs00002952.fits',
'Dark_3600.000secs00002953.fits', 'Dark_3600.000secs00002954.fits',
'Dark_3600.000secs00002955.fits', 'Dark_3600.000secs00002956.fits',
'Dark_3600.000secs00002957.fits', 'Dark_3600.000secs00002958.fits',
'Dark_3600.000secs00002959.fits', 'Flat_R_1.000secs00005390.fits',
'Flat_R_1.000secs00005391.fits', 'Flat_R_1.000secs00005392.fits',
'Flat_R_1.000secs00005393.fits', 'Flat_R_1.000secs00005394.fits',
'Flat_R_1.000secs00005395.fits', 'Flat_R_1.000secs00005396.fits',
'Flat_R_1.000secs00005397.fits', 'Flat_R_1.000secs00005398.fits',
'Flat_R_1.000secs00005399.fits', 'M51.fits', 'NGC_2997_R_00005534.fits',
'NGC_2997_R_00005535.fits', 'NGC_2997_R_00005536.fits',
'NGC_2997_R_00005537.fits', 'NGC_2997_R_00005538.fits',
'NGC_2997_R_00005617.fits', 'NGC_2997_R_00005618.fits',
'NGC_2997_R_00005619.fits', 'NGC_2997_R_00005620.fits',
'NGC_2997_R_00005621.fits']
```

> This command adds all files within the current working directory (path is ".") into an instance of the ImageFileCollection class.

STOP! Are you remembering to take notes in this jupyter notebook? Don't forget to comment on the code that we provide for you (e.g. what do the 3 lines of code above do?)

This list of images includes bias, dark and flat field images, whereas we often only want to deal with one type of image (or images taken in one filter) at a time. TheSkyX software used to control the CCDs at the Monash telescopes use the header keyword PICTTYPE to denote biases, darks, flats and science images.

Task 2. Filter the images for PICTTYPE=2. What type of images are these?

[ ]:
```
print('Printing file names')
print( images.files_filtered(PICTTYPE = 2) )

print('Printing a list of file names')
filenames = ( images.files_filtered(PICTTYPE = 2) )
print(filenames)
```

```
Printing file names
['Bias_00005290.fits' 'Bias_00005291.fits' 'Bias_00005292.fits'
 'Bias_00005293.fits' 'Bias_00005294.fits' 'Bias_00005295.fits'
 'Bias_00005296.fits' 'Bias_00005297.fits' 'Bias_00005298.fits'
 'Bias_00005299.fits']
Printing a list of file names
['Bias_00005290.fits' 'Bias_00005291.fits' 'Bias_00005292.fits'
 'Bias_00005293.fits' 'Bias_00005294.fits' 'Bias_00005295.fits'
 'Bias_00005296.fits' 'Bias_00005297.fits' 'Bias_00005298.fits'
 'Bias_00005299.fits']
```

This header would correspond to bias images.

The second line of code assigns a variable name to the filtered ImageFileCollection instance. Calling print on this writes to output the **str** representation of this object, which seems to be just the representation of the underlying array.

3 Bias

Task 3. Lets load all the bias frames with a single line of code. We will use a "list comprehension" to do so, which can make python very succinct compared to other languages. Run the following code cells. What do you think the various bits of this code are doing?

(For more information about for list comprehensions, here are a guide: https://www.digitalocean.com/community/tutorials/understanding-list-comprehensions-in-python-3)

[ ]:
```
biases = [ CCDData.read(fn, unit = "adu") for fn in images.
  ↪files_filtered(PICTTYPE = 2) ]

print(len(biases), ' bias images loaded')
```

```
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.339340
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.339340 from DATE-OBS'.
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-X to
-4130881.901 from OBSGEO-[LBH].
Set OBSGEO-Y to  2896022.315 from OBSGEO-[LBH].
Set OBSGEO-Z to -3889419.901 from OBSGEO-[LBH]'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-X to
-4130881.901 from OBSGEO-[LBH].
```

```
Set OBSGEO-Y to  2896022.315 from OBSGEO-[LBH].
Set OBSGEO-Z to -3889419.901 from OBSGEO-[LBH]'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.339351
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.339351 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.339363
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.339363 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.339374
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.339374 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.339385
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.339385 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.339397
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.339397 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.339408
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.339408 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.339420
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.339420 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.339431
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.339431 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.339442
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.339442 from DATE-OBS'.

10  bias images loaded
```

List comprehension provides syntactic sugar for iterating over an existing iterable. It will assign a name to an element from the underlying variable, then create a new iterable using an expression based on each of the elements.

Here the syntax will take each bias image and read the ADU count array for each image.

[ ]:
```python
for fn in images.files_filtered( PICTTYPE = 2 ):
    print(fn)
```

```
Bias_00005290.fits
Bias_00005291.fits
Bias_00005292.fits
Bias_00005293.fits
Bias_00005294.fits
Bias_00005295.fits
Bias_00005296.fits
Bias_00005297.fits
Bias_00005298.fits
Bias_00005299.fits
```

Note: When executing loops and functions in python, the ":" and subsequent indentation (with tab) are critical, so please keep this in mind for future work. If you want to know more about loops, take a look at https://www.w3schools.com/python/python_for_loops.asp

Task 4. Lets print some example pixel values. Have you figured out how to leave comments in a code cell? The simplest way is using a hash. Below we have added commend lines. Run the code cell and edit the comment lines to answer the following questions: What do these values correspond to? Which images and rows/coluns do these values come from?

[ ]:
```
# These values correspond to the table of ADU values for each of the bias␣
 ↪images that we have loaded before.
print(biases)
# This prints the table of ADU values for the first bias image - in this case␣
 ↪Bias_00005290.fits
print(biases[0])
# This prints the ADU values for the first column in the first bias image.
print(biases[0][:,0])
```

```
[CCDData([[2146, 2120, 2125, …, 2088, 2040, 2041],
          [2138, 2156, 2079, …, 2113, 2079, 2054],
          [2166, 2149, 2055, …, 2153, 2110, 2094],
          …,
          [2108, 2122, 2128, …, 2045, 2261, 2135],
          [2164, 2121, 2207, …, 2112, 2151, 2163],
          [2156, 2175, 2131, …, 2142, 2101, 2101]], unit='adu'),
 CCDData([[2133, 2117, 2070, …, 2111, 2059, 2077],
          [2180, 2102, 2120, …, 2206, 2130, 1993],
          [2221, 2187, 2089, …, 2008, 2129, 2096],
          …,
          [2194, 2123, 2174, …, 2151, 2163, 2073],
          [2232, 2198, 2151, …, 2125, 2114, 2045],
          [2155, 2148, 2153, …, 2060, 2123, 2078]], unit='adu'),
 CCDData([[2159, 2166, 2170, …, 2073, 2095, 2072],
          [2203, 2074, 2111, …, 2108, 2103, 2037],
          [2147, 2108, 2074, …, 2123, 2102, 2079],
          …,
          [2152, 2124, 2085, …, 2086, 2096, 2037],
          [2163, 2117, 2089, …, 2042, 2117, 2073],
```

```
         [2161, 2235, 2151, …, 2116, 2153, 2132]], unit='adu'),
CCDData([[2191, 2153, 2058, …, 2150, 2056, 2032],
         [2088, 2223, 2090, …, 2047, 2092, 2163],
         [2044, 2173, 2123, …, 2101, 2102, 2080],
         …,
         [2146, 2158, 2175, …, 2031, 2153, 2159],
         [2184, 2109, 2062, …, 2102, 2056, 2076],
         [2124, 2178, 2191, …, 2200, 2134, 2116]], unit='adu'),
CCDData([[2166, 2145, 2141, …, 2066, 2145, 2105],
         [2118, 2120, 2170, …, 2071, 2108, 2177],
         [2182, 2078, 2123, …, 2085, 2076, 2108],
         …,
         [2199, 2168, 2105, …, 2179, 2133, 2157],
         [2150, 2154, 2158, …, 2128, 2116, 2154],
         [2212, 2104, 2121, …, 2096, 2119, 2143]], unit='adu'),
CCDData([[2139, 2130, 2134, …, 2144, 2092, 2106],
         [2064, 2088, 2165, …, 2126, 2115, 2122],
         [2105, 2086, 2079, …, 2136, 2141, 2036],
         …,
         [2102, 2203, 2217, …, 2092, 2110, 2133],
         [2117, 2157, 2132, …, 2058, 2130, 2121],
         [2119, 2203, 2129, …, 2081, 2092, 2136]], unit='adu'),
CCDData([[2176, 2195, 2081, …, 2104, 2104, 2097],
         [2148, 2134, 2133, …, 2070, 2052, 2090],
         [2181, 2183, 2160, …, 2055, 2087, 2076],
         …,
         [2144, 2137, 2112, …, 2111, 2086, 2192],
         [2145, 2104, 2069, …, 2054, 2075, 2116],
         [2141, 2165, 2158, …, 2165, 2069, 2219]], unit='adu'),
CCDData([[2223, 2084, 2143, …, 2127, 2066, 2033],
         [2166, 2136, 2169, …, 2110, 2073, 2127],
         [2156, 2048, 2128, …, 2089, 2070, 2085],
         …,
         [2148, 2129, 2201, …, 2055, 2114, 2121],
         [2153, 2139, 2179, …, 2130, 2042, 2116],
         [2080, 2173, 2139, …, 2146, 2112, 2114]], unit='adu'),
CCDData([[2181, 2130, 2109, …, 2090, 2098, 2059],
         [2113, 2054, 2210, …, 2073, 2149, 2115],
         [2166, 2100, 2169, …, 2103, 2116, 2107],
         …,
         [2153, 2122, 2117, …, 2096, 2106, 2157],
         [2207, 2179, 2192, …, 2051, 2082, 2107],
         [2199, 2108, 2224, …, 2064, 2156, 2151]], unit='adu'),
CCDData([[2153, 2138, 2150, …, 2062, 2085, 2098],
         [2113, 2032, 2148, …, 2104, 2056, 2101],
         [2166, 2099, 2118, …, 2087, 2036, 2065],
         …,
         [2186, 2135, 2087, …, 2158, 2073, 2161],
```

```
         [2113, 2126, 2108, …, 2124, 2164, 2132],
         [2126, 2155, 2105, …, 2067, 2068, 2083]], unit='adu')]
[[2146 2120 2125 … 2088 2040 2041]
 [2138 2156 2079 … 2113 2079 2054]
 [2166 2149 2055 … 2153 2110 2094]
 …
 [2108 2122 2128 … 2045 2261 2135]
 [2164 2121 2207 … 2112 2151 2163]
 [2156 2175 2131 … 2142 2101 2101]] adu
[2146 2138 2166 … 2108 2164 2156] adu
```

Task 5. Print the count statistics (mean, stdev, min & max) for the first bias image, modifying the code provided below.

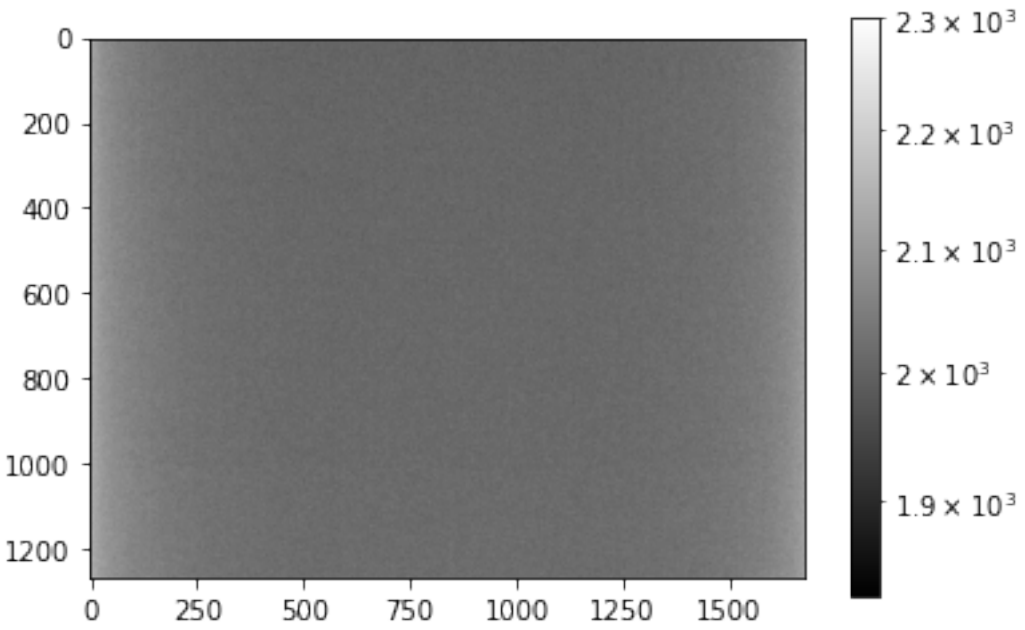
[ ]:
```
print('Min:', np.min(biases[0]))
print('Max:', np.max(biases[0]))
print('Mean:', np.mean(biases[0]))
print('Stdev:', np.std(biases[0]))


print('\nMedian:', np.median(biases[0]))
```

```
Min: 1829
Max: 2301
Mean: 2023.3931362178553
Stdev: 43.27904831213284

Median: 2021.0
```

Task 6. Display the first bias image below.

[ ]:
```
from matplotlib.colors import LogNorm
plt.imshow(biases[0], cmap='gray', norm=LogNorm())
plt.colorbar()
```

[ ]:
```
<matplotlib.colorbar.Colorbar at 0x1651e6b30>
```



Task 7. Like in ds9, we can look at the header keywords. Run the following cell code to display the header of the 1st bias image

[ ]:
```
print(biases[0].header)
```

```
SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                   16 / number of bits per data pixel
NAXIS   =                    2 / number of data axes
NAXIS1  =                 1679 / length of data axis 1
NAXIS2  =                 1268 / length of data axis 2
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in
'AstronomyCOMMENT   and Astrophysics', volume 376, page 359; bibcode:
2001A&A…376..359H BZERO   =                32768 / offset data range to that
of unsigned short    BSCALE  =                    1 / default scaling factor
OBSERVER= 'Monash C14 Observer' / SBIGFITSEXT The name of the observer
ORIGIN  = 'Monash C14'         / Description of location
TELESCOP= 'C14     '           / SBIGFITSEXT The model Telescope
FOCALLEN=               3910. / SBIGFITSEXT Telescope focal length in mm
APTDIA  =               355.6 / SBIGFITSEXT Aperture diameter in mm
SBUUID  = '{51a71438-c455-462e-9958-13758eb8181f}' / Photo UUID
EXPTIME =                  0. / SBIGFITSEXT Total exposure time in seconds
SWCREATE= 'TheSkyX Version 10.5.0 Build 12503' / SBIGFITSEXT Name & version of
sCOLORCCD=                   0 / Non zero if image is from a Bayer color ccd
DISPINCR=                    1 / Non zero to automatically display the image in
PICTTYPE=                    2 / Image type as index 0= Unknown 1=Light,
```

```
2=Bias,IMAGETYP= 'Bias Frame'         / SBIGFITSEXT Light, Dark, Bias or Flat
XORGSUBF=                    0 / SBIGFITSEXT Subframe x upper-left pixel in bin
YORGSUBF=                    0 / SBIGFITSEXT Subframe y upper-left pixel in bin
XBINNING=                    2 / SBIGFITSEXT Binning factor in width
YBINNING=                    2 / SBIGFITSEXT Binning factor in height
EXPSTATE=                  294
EGAIN   =                 0.37 / SBIGFITSEXT Electronic gain in e- per ADU
CCD-TEMP=    -20.2797644771323 / SBIGFITSEXT Temperature of the CCD
SET-TEMP=    -20.5004215411525 / SBIGFITSEXT The cooler setpoint in degrees C
SITELAT = '-37 49 01.20'       / SBIGFITSEXT Latitude of the imaging location
SITELONG= '-144 58 01.18'      / SBIGFITSEXT Longitude of the imaging location
LST     = '+06 32 05.00'       / Local sidereal time
OBSGEO-B=    -37.8170013427734 / Latitude of the observation in degrees, North
+OBSGEO-L=     144.966995239258 / Longitude of the observation in degrees, East
+OBSGEO-H=                   0. / Altitude of the observation in meters
BTP     =                    1 / Beyond the pole
CENTAZ  =     39.7689602283439 / SBIGFITSEXT Azimuth of the center of the image
CENTALT =     42.4347084876392 / SBIGFITSEXT Altitude of the center of the
imageAIRMASS =     1.48199842775995 / Airmass of the telescope
TELEHA  = '-01 52 45.70'       / Telescope hour angle
MOUNT   = 'Paramount MX'       / The telescope mount
OBJCTRA = '08 23 45.111'       / SBIGFITSEXT The right ascension of the center
oOBJCTDEC= '+02 02 33.35'       / SBIGFITSEXT The declination of the center of
thOBJECT  = 'M 50    '           / SBIGFITSEXT The name of the object imaged
INSTRUME= 'SBIG STT-8300 3 CCD Camera' / SBIGFITSEXT The model camera used.
XPIXSZ  =                 10.8 / SBIGFITSEXT Pixel width in microns after
binninYPIXSZ  =                 10.8 / SBIGFITSEXT Pixel height in microns after
binniPEDESTAL=                -100 / SBIGFITSEXT Add this count to each pixel
value FOCPOS  =                1271. / The focuser position.
FOCTEMP =                  100. / The focuser temperature in degrees C.
FOCTMPSC= 'Focuser (Default)'  / The focuser temperature source.
FILTER  = 'U       '           / SBIGFITSEXT The optical filter used to take
imaDATE-OBS= '2021-04-02T08:08:38.988' / SBIGFITSEXT UTC of start exp. in ISO
8601 LOCALTIM= '2/04/2021 07:08:38.990 PM STD' / Local time at exposure start
END
```

Task 8. Let look at just one header keyword. What is this keyword and its unit?

[ ]:
```
print(biases[0].header['EXPTIME'])
```

```
0.0
```

This displays the total exposure time in seconds. As expected, it's 0.0 for a bias image.

Task 9. To improve the signal-to-noise of images and mitigate spurious signals (such as radiation hits) we combine images together. A median combine is relatively simple and automatically rejects spuriously high and low values that occur in just a few individual images.

A good habit to get into is to print image statistics and see if they behave as you would expect. Is this the case here?

[ ]:
```
bias_median = ccdproc.Combiner(biases, dtype=np.float32).median_combine()

# A good habit to get into is printing the statistics of input and output images
print('Image statistics for the median bias')

thisimage=bias_median
print('Min:', np.min(thisimage))
print('Max:', np.max(thisimage))
print('Median:', np.median(thisimage))
print('Std Dev:', np.std(thisimage))
```

```
Image statistics for the median bias
Min: 1943.5
Max: 2205.0
Median: 2017.5
Std Dev: 26.5631046295166
```

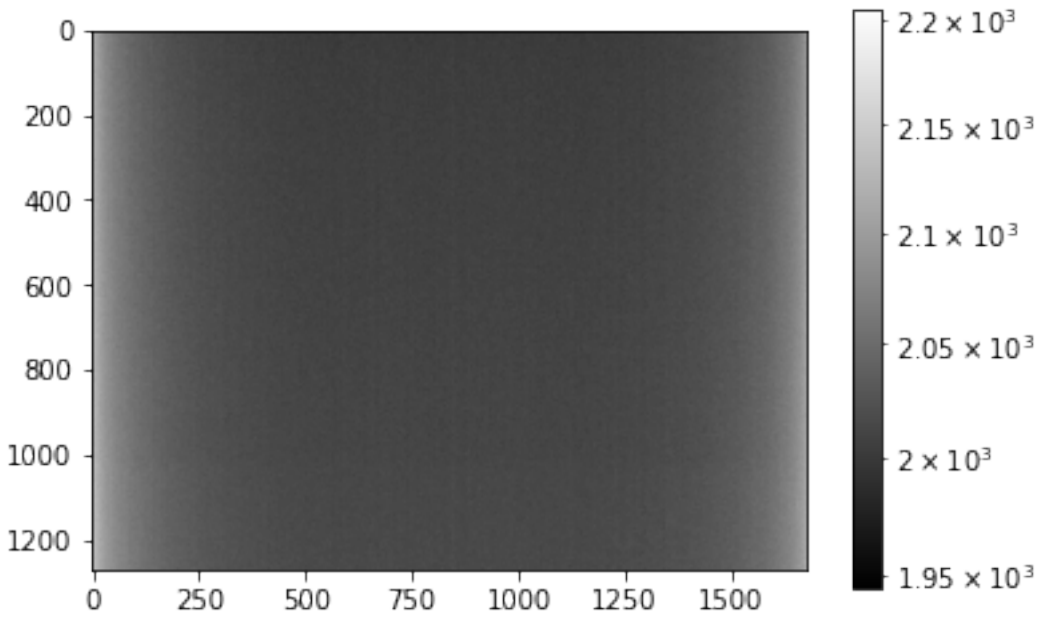
```
/Users/ced/.local/share/virtualenvs/ASP3231_pipenv-dMPBVrY6/lib/python3.10/site-
packages/numpy/core/fromnumeric.py:758: UserWarning: Warning: 'partition' will
ignore the 'mask' of the MaskedArray.
  a.partition(kth, axis=axis, kind=kind, order=order)
```

This seems reasonable. The minimum, maximum and median values are similar to when we inspected the biases[0] file individually, but the std deviation has understandably decreased as a result of combining the images.

Task 10. Display this new image, print some example pixel values and calculate some basic statistics. How does this compare with the individual exposure you tried earlier?

[ ]:
```
from matplotlib.colors import LogNorm
plt.imshow(thisimage, cmap='gray', norm=LogNorm())
plt.colorbar()
```

[ ]:
```
<matplotlib.colorbar.Colorbar at 0x28800e2c0>
```



Task 11. Lets print the header information

```
[ ]: print(bias_median.header)
```

```
OrderedDict([('NCOMBINE', 10)])
```

Task 12. Unfortunately some useful keywords are missing, so lets add them to the header manually.

```
[ ]: bias_median.meta.update(EXPTIME = 0)
bias_median.meta.update(TELESCOP = biases[0].header['TELESCOP'])
bias_median.meta.update(OBJECT = 'Bias_Median')
print(bias_median.meta)
```

```
OrderedDict([('NCOMBINE', 10), ('EXPTIME', 0), ('TELESCOP', 'C14'), ('OBJECT',
'Bias_Median')])
```

Task 13. If you are happy with the combined image than you can write it to an output FITS file.

```
[ ]: bias_median.write("bias_median.fits")
```

Task 14. We can delete the individual bias data from the notebook (the raw bias FITS files will remain on disk) and use garbage collect (gc) to clear memory.

```
[ ]: del(biases)
collected = gc.collect()
print('Check garbage collection', collected)
```

```
Check garbage collection 8856
```

4 Dark

We now need to produce an image of the dark current, as the dark current can vary from pixel to pixel. To do this we must identify the dark images, subtract the bias from them and then combine the processed dark frames together.

Task 15. The first steps are very similar to those we undertook for the bias frames. Run the following cell code. What are each of these lines doing? How do these lines differ from what you used for the bias frames?

> Here, the first line is analogous, and uses a wildcard regex to match all the dark files. In that sense, we're creating the ImageFileCollection directly with only the dark images, instead of creating it with every file and filtering it by metadata.
>
> We then perform the filtering for PICTTYPE = 3 - although in theory if all the naming convention and metadata assignment has been done correclty, nothing should be filtered out here. But it's good just to be sure.
>
> Then, we're reading the adu data for each file as before.

[ ]:
```
images = ccdproc.ImageFileCollection(".",glob_include = 'Dark_*')
for fn in images.files_filtered(PICTTYPE = 3):
    print(fn)
darks = [ CCDData.read(fn, unit = "adu") for fn in images.
 ↪files_filtered(PICTTYPE = 3) ]
```

```
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59295.224946
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59295.224946 from DATE-OBS'.
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-X to
-4130881.901 from OBSGEO-[LBH].
Set OBSGEO-Y to  2896022.315 from OBSGEO-[LBH].
Set OBSGEO-Z to -3889419.901 from OBSGEO-[LBH]'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-X to
-4130881.901 from OBSGEO-[LBH].
Set OBSGEO-Y to  2896022.315 from OBSGEO-[LBH].
Set OBSGEO-Z to -3889419.901 from OBSGEO-[LBH]'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59295.266624
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59295.266624 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59295.308302
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59295.308302 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59295.349980
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59295.349980 from DATE-OBS'.
```

```
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59295.391659
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59295.391659 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59295.433337
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59295.433337 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59295.475015
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59295.475015 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59295.516693
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59295.516693 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59295.558372
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59295.558372 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59295.600050
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59295.600050 from DATE-OBS'.

Dark_3600.000secs00002950.fits
Dark_3600.000secs00002951.fits
Dark_3600.000secs00002952.fits
Dark_3600.000secs00002953.fits
Dark_3600.000secs00002954.fits
Dark_3600.000secs00002955.fits
Dark_3600.000secs00002956.fits
Dark_3600.000secs00002957.fits
Dark_3600.000secs00002958.fits
Dark_3600.000secs00002959.fits
```

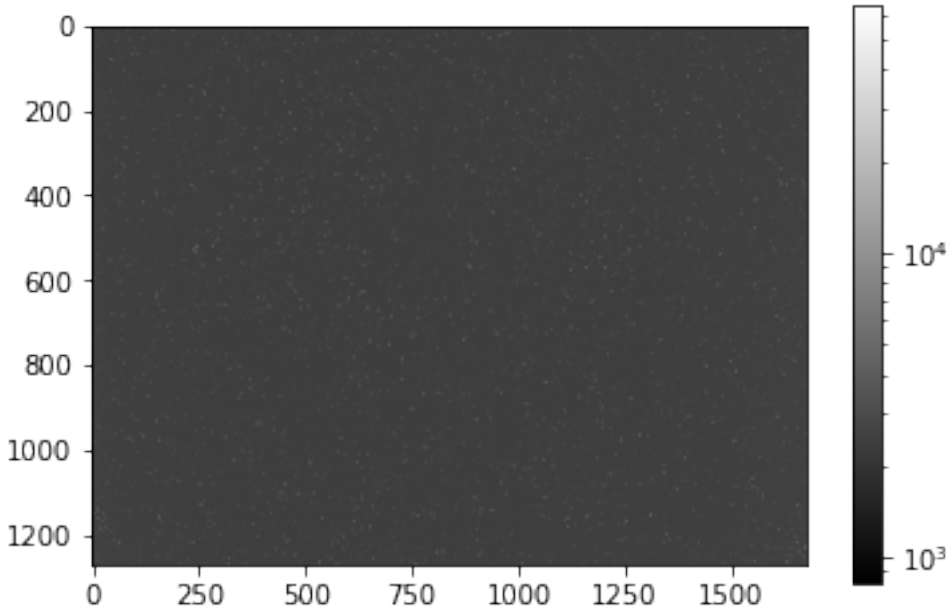
Task 16. Print some example pixel values, calculate some basic statistics for one dark image, and display one dark image. To do this you can copy & paste the commands you used for the bias images, and *modify* them accordingly. When using jupyter notebooks preserve the commands you have used in order so they can be run in sequence to reproduce your work.

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

print('Min: ', np.min(darks[0]))
print('Max: ', np.max(darks[0]))
print('Mean: ', np.mean(darks[0]))
print('Median: ', np.median(darks[0]))
```

```
print('Stdev: ', np.std(darks[0]))

plt.imshow(darks[0], cmap='gray', norm=LogNorm())
plt.colorbar()
```

```
[[2539 2433 2561 … 2446 2375 2399]
 [2600 2715 2553 … 2486 2451 2485]
 [2484 2474 2584 … 2453 2390 2440]
 …
 [2564 2622 2567 … 2552 2491 2620]
 [2609 2549 2559 … 2537 2432 2609]
 [2612 2566 2508 … 2526 2577 2476]] adu
[2539 2600 2484 … 2564 2609 2612] adu
Min:  815
Max:  65535
Mean:  2452.3977008622
Median:  2396.0
Stdev:  1010.9193488546595
```

[ ]: <matplotlib.colorbar.Colorbar at 0x29b176d10>



Note here that there seems to be a lot of hot pixels in the darks - we can see this is first indicated by the fact that the constraints of the generated LogNorm colourbar spans orders of magnitude, compared to before with the flats. This too is reflected in the fact that the stdev is high, and that the max is 65536 - there are a LOT of hot pixels.

Task 17. CCDproc has a command for bias subtraction so let's now use it on the individual dark

exposures (be warned it is possible to run this command twice and over-subtract).

[ ]:
```
# for idx, thisimage in enumerate(darks):
#     darks[idx] = ccdproc.subtract_bias(thisimage, bias_median)

unbiased_darks = [ccdproc.subtract_bias(dark, bias_median) for dark in darks]
```

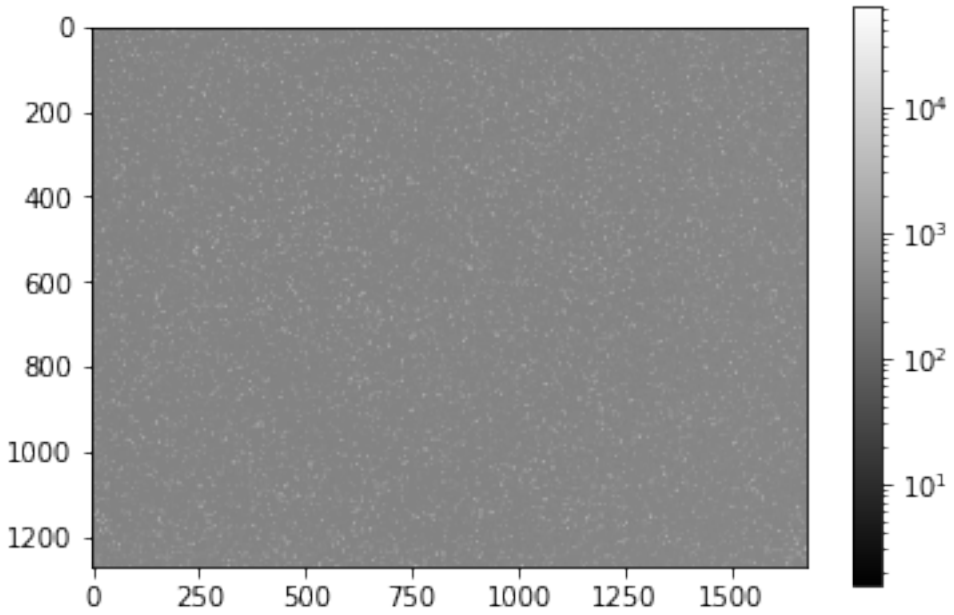
I've modified the code to introduce a new variable name for the darks with biases subtracted.

Task 18. Print some example pixel values, statistics and display an image. How does this all compare with what you determined for an individual dark exposure earlier? Does the change reflect what you predict in your pre-lab (Task 2)?

[ ]:
```
print(unbiased_darks[0])
print('Min: ', np.min(unbiased_darks[0]))
print('Max: ', np.max(unbiased_darks[0]))
print('Mean: ', np.mean(unbiased_darks[0]))
print('Median: ', np.median(unbiased_darks[0]))
print('Stdev: ', np.std(unbiased_darks[0]))

plt.imshow(unbiased_darks[0], cmap='gray', norm=LogNorm())
plt.colorbar()
```

```
[[376.5 299.  431.5 … 349.  286.5 324.5]
 [472.  604.  412.5 … 380.  353.5 377. ]
 [318.  370.  463.5 … 358.  288.  357.5]
 …
 [414.  490.  444.5 … 458.  379.  474. ]
 [451.  416.5 417.5 … 430.  317.  493. ]
 [464.  397.  363.  … 420.  461.5 352. ]] adu
Min:  -1196.5
Max:  63555.5
Mean:  429.9424839781829
Median:  374.0
Stdev:  1010.6946936371945
```

[ ]:
```
<matplotlib.colorbar.Colorbar at 0x29b2b9f00>
```



Task 19. Combine the processed dark exposures together, using the following command.

[ ]:
```
unbiased_dark_median = ccdproc.Combiner(unbiased_darks, dtype=np.float32).
↪median_combine()
```

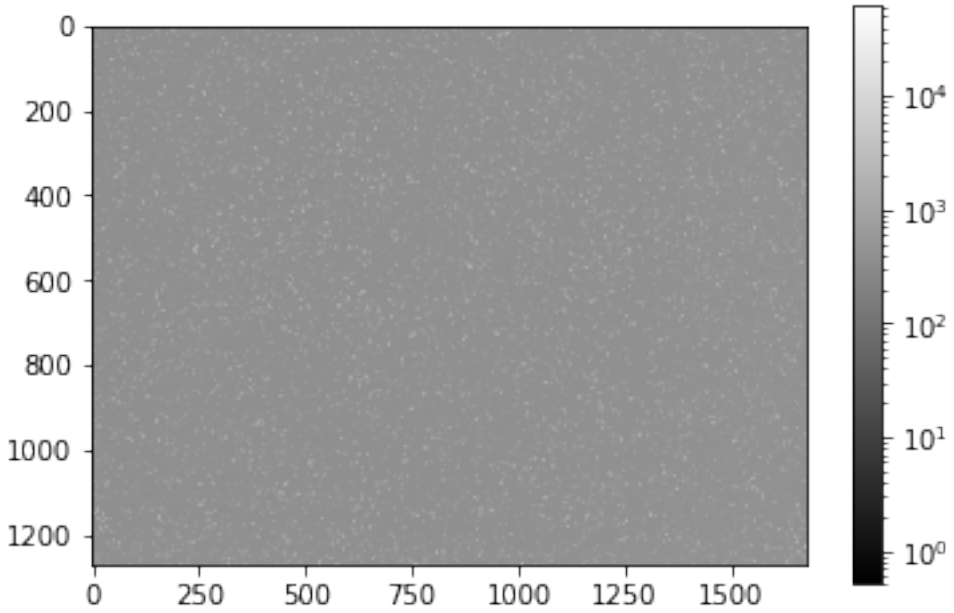
Task 20. Print pixel values, statistics and display dark_median. How do these values values compare with what's measured for individual exposures?

[ ]:
```
print(unbiased_dark_median)
print('Min: ', np.min(unbiased_dark_median))
print('Max: ', np.max(unbiased_dark_median))
print('Mean: ', np.mean(unbiased_dark_median))
print('Median: ', np.median(unbiased_dark_median))
print('Stdev: ', np.std(unbiased_dark_median))

plt.imshow(unbiased_dark_median, cmap='gray', norm=LogNorm())
plt.colorbar()
```

```
[[341.5 358.5 355.  … 349.  317.  334. ]
 [464.  496.5 410.5 … 378.5 386.5 379.5]
 [347.5 395.  399.5 … 356.  350.5 365.5]
 …
 [469.5 444.  374.5 … 423.  361.5 462.5]
 [444.5 355.5 415.  … 510.5 413.  484. ]
 [436.  359.5 396.  … 424.  408.  428.5]] adu
Min:  -1203.5
Max:  63555.5
```

```
Mean:  428.23505
Median:  373.0
Stdev:  1003.1648559570312
```

[ ]:
```
<matplotlib.colorbar.Colorbar at 0x2c04bb7c0>
```



Here, we have a mean that's closer to 0, which is more reasonable. There are weird circular regions of increased detection though, and these regions seem to have remained after the calculation of the median, which seems to indicate to me that the pixel locations of these circular objects are constant across all the dark images. I'm not sure what these are.

Task 21. Print the header.

[ ]:
```
print(unbiased_dark_median.header)
```

```
OrderedDict([('NCOMBINE', 10)])
```

Task 22. Again, we have missing keywords, so we will add them manually and write the output image.

[ ]:
```
unbiased_dark_median.meta.update(EXPTIME = 3600)
unbiased_dark_median.meta.update(TELESCOP = 'C14')
unbiased_dark_median.meta.update(OBJECT = 'Dark_Median')
unbiased_dark_median.write("dark_median.fits")
```

Task 23. If you're happy with dark_median then delete the individual dark exposures (the raw FITS files will remain) and use gc to clear memory.

[ ]:
```
del(darks)
del(unbiased_darks)
collected = gc.collect()
print('Check garbage collection', collected)
```

```
Check garbage collection 3127
```

5 Flats

Unsurprisingly, the next step is identifying the loading the flat-field images, processing them (using the bias and dark images) and then producing a median combined flat.

Task 24. Run the following code cells.

[ ]:
```
images = ccdproc.ImageFileCollection(".")
for fn in images.files_filtered(PICTTYPE = 4):
    print(fn)
```

```
Flat_R_1.000secs00005390.fits
Flat_R_1.000secs00005391.fits
Flat_R_1.000secs00005392.fits
Flat_R_1.000secs00005393.fits
Flat_R_1.000secs00005394.fits
Flat_R_1.000secs00005395.fits
Flat_R_1.000secs00005396.fits
Flat_R_1.000secs00005397.fits
Flat_R_1.000secs00005398.fits
Flat_R_1.000secs00005399.fits
```

But wait! Each filter has a different flat-field image, so we have to deal with each filter individually. Good naming conventions help. For this lab, we will only deal with the R-band flats.

[ ]:
```
images = ccdproc.ImageFileCollection(".",glob_include = 'Flat_R_*')
for fn in images.files_filtered(PICTTYPE = 4):
    print(fn)
flats = [ CCDData.read(fn, unit = "adu") for fn in images.
  ↪files_filtered(PICTTYPE = 4) ]
```

```
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.355416
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.355416 from DATE-OBS'.
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-X to
-4130881.901 from OBSGEO-[LBH].
Set OBSGEO-Y to  2896022.315 from OBSGEO-[LBH].
Set OBSGEO-Z to -3889419.901 from OBSGEO-[LBH]'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-X to
-4130881.901 from OBSGEO-[LBH].
Set OBSGEO-Y to  2896022.315 from OBSGEO-[LBH].
```

```
Set OBSGEO-Z to -3889419.901 from OBSGEO-[LBH]'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.355442
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.355442 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.355467
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.355467 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.355491
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.355491 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.355516
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.355516 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.355541
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.355541 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.355565
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.355565 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.355590
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.355590 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.355616
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.355616 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.355641
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.355641 from DATE-OBS'.

Flat_R_1.000secs00005390.fits
Flat_R_1.000secs00005391.fits
Flat_R_1.000secs00005392.fits
Flat_R_1.000secs00005393.fits
Flat_R_1.000secs00005394.fits
Flat_R_1.000secs00005395.fits
Flat_R_1.000secs00005396.fits
Flat_R_1.000secs00005397.fits
Flat_R_1.000secs00005398.fits
Flat_R_1.000secs00005399.fits
```

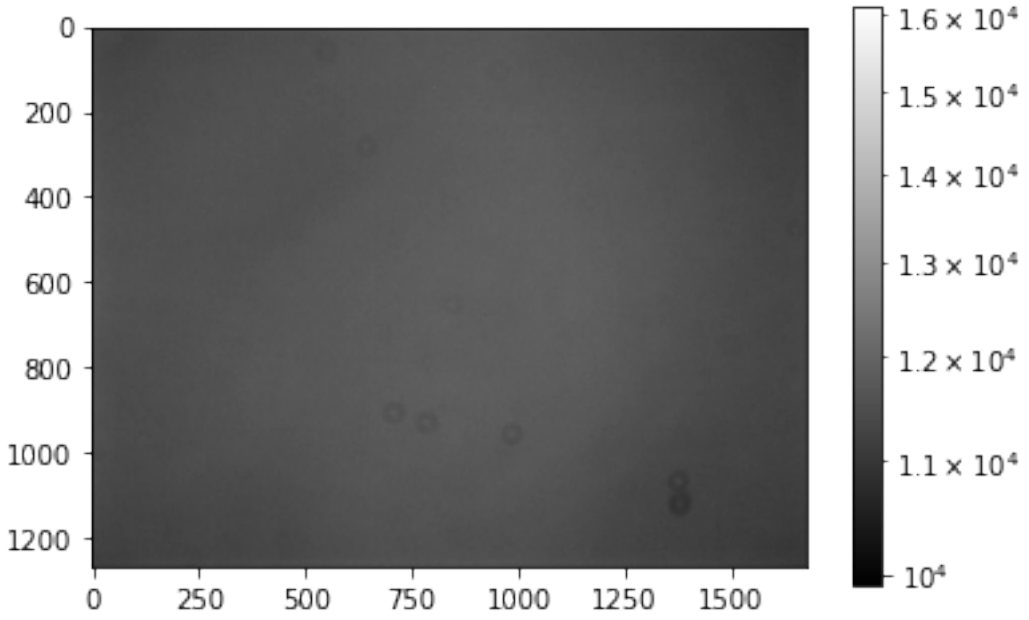
Task 25. Print the counts of the first column of the first flat image and display one flate image and describe the features that you see. Also, open the fits file in ds9 to see more details.

[ ]:
```
def print_stats(image):
    print('Min: ', np.min(image))
    print('Max: ', np.max(image))
    print('Mean: ', np.mean(image))
    print('Median: ', np.median(image))
    print('Stdev: ', np.std(image))
```

[ ]:
```
print(flats[0][:,0])
print_stats(flats[0])

plt.imshow(flats[0], cmap='gray', norm=LogNorm())
plt.colorbar()
```

```
[11858 11435 11065 … 11176 10912 10413] adu
Min:  9895
Max:  16124
Mean:  11565.56693653087
Median:  11575.0
Stdev:  257.4639838332078
```

[ ]: `<matplotlib.colorbar.Colorbar at 0x287dee950>`



Note here that the standard deviation is quite low and the colourbar is constrained well within an order of magnitude. As such, the observations here are quite uniform.

We also see these dark annular objects in the lower region of the image - this could be the shadow of out of focus dust particles on the glass panel in front of the detector.

Task 26. Let's now process the individual flat field frames using the bias and dark images. After each step, print out the count of the first column of the first flat image (we have left the blank print statements in the code cell). Notice the dark subtraction has a dependence on exposure time. Why is this the case? How did the pixel values change during each data reduction step? Does this meet your expectations?

[ ]:
```
for idx, thisimage in enumerate(flats):
    flats[idx] = ccdproc.subtract_bias(thisimage, bias_median)
print(flats[0][:,0])

for idx, thisimage in enumerate(flats):
    flats[idx] = ccdproc.subtract_dark(thisimage, unbiased_dark_median,␣
 ↪exposure_time = 'EXPTIME',
                                       exposure_unit = u.second, scale = True)
print(flats[0][:,0])
```

```
[9695.5 9307.  8899.  … 9026.  8754.  8265. ] adu
[9695.40513889 9306.87111111 8898.90347222 … 9025.86958332 8753.87652777
 8264.87888888] adu
```

Here we've performed successive subtractions of the bias and the dark. The dark subtraction has a dependence on exposure time since the effect of the error associated with the dark in the flat image is dependent on how long the exposure time of the flat is. The pixel values have decreased in a predictable way.

[ ]:
```
print_stats(flats[0])
```

```
Min:  7801.898333333433
Max:  13987.88430595398
Mean:  9542.992765453922
Median:  9555.374374993145
Stdev:  268.1342519182293
```

```
/Users/ced/.local/share/virtualenvs/ASP3231_pipenv-dMPBVrY6/lib/python3.10/site-
packages/numpy/core/fromnumeric.py:758: UserWarning: Warning: 'partition' will
ignore the 'mask' of the MaskedArray.
  a.partition(kth, axis=axis, kind=kind, order=order)
```

Task 27. Lets now have a look at the median pixel count values of individual flat field exposures. Are the counts behaving as you'd expect and why?

[ ]:
```
for thisimage in flats:
    print('Median:', np.ma.median(thisimage.data))
```

```
Median: 9555.374374993145
Median: 9467.415902774781
Median: 9378.409583330154
```

```
Median: 9298.890416666865
Median: 9210.397222220898
Median: 9123.899722218513
Median: 9045.41208332777
Median: 8962.384722217917
Median: 8875.892916664481
Median: 8792.888541664928
```

They seem reasonable, but they are decreasingly for each successive flat field exposure. If these flat field exposures were taken in the twilight sky, this could be as a result of decreasing light as the sun sets. Let's check our hypothesis:

[ ]:
```
for flat in flats:
    print(flat.header['LOCALTIM'])
```

```
2/04/2021 07:31:47.950 PM STD
2/04/2021 07:31:50.200 PM STD
2/04/2021 07:31:52.320 PM STD
2/04/2021 07:31:54.450 PM STD
2/04/2021 07:31:56.579 PM STD
2/04/2021 07:31:58.710 PM STD
2/04/2021 07:32:00.830 PM STD
2/04/2021 07:32:02.960 PM STD
2/04/2021 07:32:05.190 PM STD
2/04/2021 07:32:07.389 PM STD
```

I'm not sure if the solar photon contribution has decreased enough over the course of 20 seconds to cause this reasonably substantial decrease in median value, but if I had to speculate, that would be why.

Task 28. What mathematical operation do we do when we flatfield correct our data? What should the typical pixel value be for a final flatfield image? Look at the lines of code below and describe what they are doing.

[ ]:
```
tempimages = flats.copy()
for idx, thisimage in enumerate(tempimages):
    m = 1.0 / np.ma.median(tempimages[idx])
    tempimages[idx] = tempimages[idx].multiply(m * u.adu)
FlatV_median = ccdproc.Combiner(tempimages, dtype=np.float32).median_combine()
```

We're renormalising and combining the flats to create a median. The first line creates a shallow copy upon which we can operate. The loop iterates and creates a masked array of the reciprocal of the median for that image. We then reassign the reciprocal of the median by the data of the image itself.

By renormalising, we're making each pixel value a coefficient relative to the median - if the pixel is equal to the median, the pixel value should be 1. If the pixel is double the median, the pixel value will be 2.

In short, we're multiplying each of the flats by the reciprocal of its own median, then combine it using ccdproc.Combiner to create a median for all the flats.

[ ]:
```
print_stats(FlatV_median)
```

```
Min:  0.84136933
Max:  1.4834263
Mean:  0.99870163
Median:  1.0007373094558716
Stdev:  0.022937433794140816
```

```
/Users/ced/.local/share/virtualenvs/ASP3231_pipenv-dMPBVrY6/lib/python3.10/site-
packages/numpy/core/fromnumeric.py:758: UserWarning: Warning: 'partition' will
ignore the 'mask' of the MaskedArray.
  a.partition(kth, axis=axis, kind=kind, order=order)
```

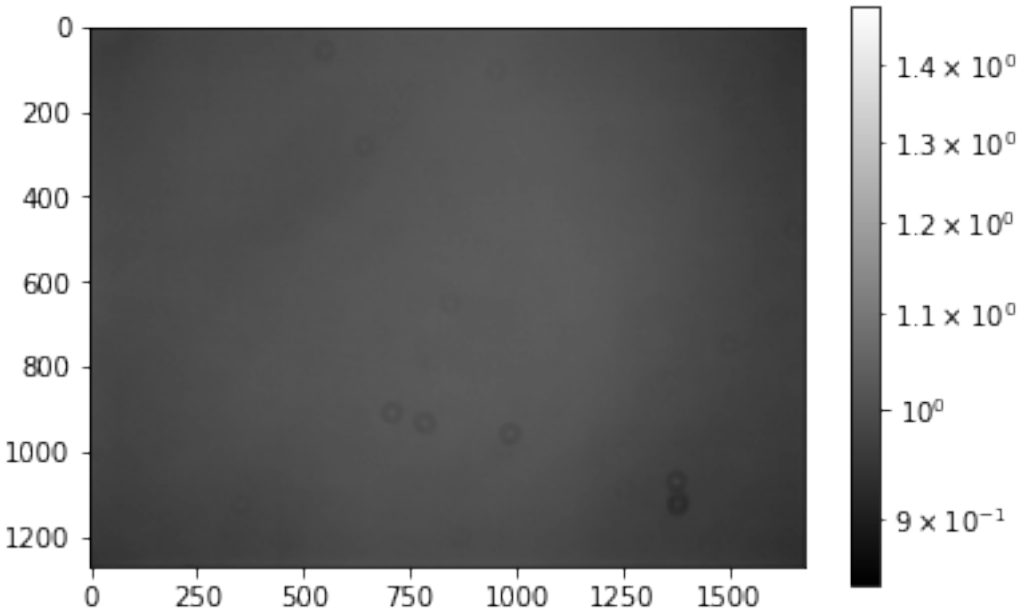
We can see our mean and medians are very close to 1, which is what we expect for our flats.

Task 29. Display Flat_V median, print some example pixel values and do the basic statistics. Is the RMS for this image dominated by random fluctuations (noise) in the pixel values, by structure in the image or are you unsure? (If you're unsure do flag this in your notebook, and consider this before next week.)

[ ]:
```
print(FlatV_median)
print_stats(FlatV_median)

plt.imshow(FlatV_median, cmap='gray', norm=LogNorm())
plt.colorbar()
```

```
[[0.9582555  0.96436393 0.9799832  … 0.957518   0.91782033 0.86665624]
 [0.96564096 0.97469425 0.9731922  … 0.94963944 0.90339077 0.86589307]
 [0.95621896 0.9616722  0.96572363 … 0.9453608  0.8888645  0.872069  ]
 …
 [0.9571819  0.95650935 0.96432155 … 0.9554485  0.92078644 0.8704841 ]
 [0.9169915  0.90967196 0.91467106 … 0.89811647 0.8937492  0.88375014]
 [0.87593704 0.8792957  0.8757031  … 0.86000377 0.868693   0.8861635 ]] adu2
Min:  0.84136933
Max:  1.4834263
Mean:  0.99870163
Median:  1.0007373094558716
Stdev:  0.022937433794140816
```

[ ]:
```
<matplotlib.colorbar.Colorbar at 0x288145960>
```



I've plotted the Flat_V median - this doesn't really have any real meaning since the units are technically in adus, but setting the scale to lognorm gives us at least a qualitative inspection of which areas are 'overweighted'.

I would expect the RMS in this image to be dominated by the the structures in the image, ie the vignetting and the donuts.

Task 30. Add the missing metadata and save the image.

[ ]:
```
FlatV_median.meta.update(EXPTIME = 1)
FlatV_median.meta.update(TELESCOP = 'C14')
FlatV_median.meta.update(OBJECT = 'Flat_R_Median')
FlatV_median.write("Flat_R_median.fits")
```

Task 31. If you think everything has worked, then delete the individual flat exposures and use gc to clear memory.

[ ]:
```
del(flats)
del(tempimages)
collected = gc.collect()
print('Check garbage collection', collected)
```

```
Check garbage collection 29121
```

6 Reducing Science Images

6.1 Now lets reduce some V-band science images, using the bias, dark and V-band flat we have created.

Task 32. First step is to load the science images. How is the following code loading V-band science images (and not other data)?

[ ]:
```
images = ccdproc.ImageFileCollection(".",glob_include = '*_R_*')
for fn in images.files_filtered(PICTTYPE = 1):
    print(fn)
scim = [CCDData.read(fn, unit = "adu") for fn in images.files_filtered(PICTTYPE␣
 ↪= 1)]
print(scim[0])
```

```
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.414130
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.414130 from DATE-OBS'.
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-X to
-4130881.901 from OBSGEO-[LBH].
Set OBSGEO-Y to  2896022.315 from OBSGEO-[LBH].
Set OBSGEO-Z to -3889419.901 from OBSGEO-[LBH]'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-X to
-4130881.901 from OBSGEO-[LBH].
Set OBSGEO-Y to  2896022.315 from OBSGEO-[LBH].
Set OBSGEO-Z to -3889419.901 from OBSGEO-[LBH]'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.415536
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.415536 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.416942
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.416942 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.418346
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.418346 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.419752
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.419752 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.446194
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.446194 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.447599
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
```

```
59306.447599 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.449004
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.449004 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.450409
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.450409 from DATE-OBS'.
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59306.451815
from DATE-OBS'. [astropy.wcs.wcs]
WARNING:astropy:FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to
59306.451815 from DATE-OBS'.

NGC_2997_R_00005534.fits
NGC_2997_R_00005535.fits
NGC_2997_R_00005536.fits
NGC_2997_R_00005537.fits
NGC_2997_R_00005538.fits
NGC_2997_R_00005617.fits
NGC_2997_R_00005618.fits
NGC_2997_R_00005619.fits
NGC_2997_R_00005620.fits
NGC_2997_R_00005621.fits
[[ 4318  4204  4327 …  3970  3952  3921]
 [ 4260  4259  4300 …  4079  4061  3904]
 [ 4199  4194  4063 …  4163  4042  4008]
 …
 [ 4255  4371  4425 …  5912  7884 11427]
 [ 3994  4374  4065 …  6292  9081 15447]
 [ 4114  4070  4061 …  6611 10157 19818]] adu
```

This loads the science images by once again using a wildcard regex, although there is also a secondary filter by PICTTYPE metadata.

Task 33. Now we process these images using the bias, dark and V-band flat. Compare the count values of the 1st column of the first image by using the print statements. What did these steps do? What are the relevant mathematical operations? If you're unsure see Section 4.5 of Rieke, "Measuring the Universe."

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

for idx, thisimage in enumerate(scim):
    scim[idx] = ccdproc.subtract_bias(thisimage, bias_median)
print(scim[0][:,0])

for idx, thisimage in enumerate(scim):
    scim[idx] = ccdproc.subtract_dark(thisimage, unbiased_dark_median,␣
 ↪exposure_time = 'EXPTIME',
```

```
                                        exposure_unit = u.second, scale = True)
print(scim[0][:,0])

for idx, thisimage in enumerate(scim):
    scim[idx] = ccdproc.flat_correct(thisimage, FlatV_median)
print(scim[0][:,0])
```

```
[4318 4260 4199 … 4255 3994 4114] adu
[2155.5 2132.  2033.  … 2105.  1836.  1966. ] adu
[2144.11666584 2116.53333282 2021.41666603 … 2089.34999943 1821.18333244
 1951.46666622] adu
[2234.6156358  2188.99711592 2111.22374907 … 2179.97996981 1983.46309318
 2224.9692506 ] adu
```

We're first subtracting the bias, then the adjusted darks (scaled for exposure time), then finally dividing through by the flat images.

Task 34. Let's now write these images to output files, adding the proc prefix so we don't confuse processed images with raw images.

[ ]:
```
newname = []
for fn in images.files_filtered(PICTTYPE = 1):
    newname.extend(["proc_" + fn])

print(newname)

# This image writes the science images out (while including some extra lines to␣
 ↪reduce the file size)
for idx, thisimage in enumerate(scim):
    tempimages = [thisimage]
    temp = ccdproc.Combiner(tempimages,dtype=np.float32).median_combine()
    temp.meta = thisimage.meta
    temp.write(newname[idx])
```

```
['proc_NGC_2997_R_00005534.fits', 'proc_NGC_2997_R_00005535.fits',
'proc_NGC_2997_R_00005536.fits', 'proc_NGC_2997_R_00005537.fits',
'proc_NGC_2997_R_00005538.fits', 'proc_NGC_2997_R_00005617.fits',
'proc_NGC_2997_R_00005618.fits', 'proc_NGC_2997_R_00005619.fits',
'proc_NGC_2997_R_00005620.fits', 'proc_NGC_2997_R_00005621.fits']
```

Task 35. You can use ds9 to display a raw science and the processed version of that image. Do that now and compare the images. Record what you see in this notebook. Are there any images you would *not* use for your science? If so, explain why.

I opened the images in ds9 and compared them. The biggest difference is that artifacts are removed in the processed images - the dust specks (which appear in the top half of the image in ds9) are almost completely absent in the processed images. The effect of the vignetting is also gone.

The first set of images (55___) seem to be weirdly out of focus, and some of the stars towards the centre of the image appear more like disc-like bacteria under a microscope rather than star point-sources. I'm included to use the second set of images.

Task 36. Your edited notebook should now function as a piece of code, that can go through all the steps of this lab (although it will get into trouble if it tries to overwrite images.) Move your median bias, dark and flat images, and your processed science images, to a new directory. In the Kernal window select "Restart and Clear output" and then in Cell menu selection "Run All." What happens?

It will not run without the source images in the directory, but it will generate the processed images if only the source images are in the directory.

7 Conclusions

Once you have completed your notebook, please save a copy for yourself and submit it via Moodle.

Task 37. This is optional, but feedback will help the TAs adjust the labs and questions according to the needs of the class. How long did it take you to complete it lab? Was this lab difficult? Are there any changes you would like to see that can improves your experience in this unit?

[ ]: