Scaling & Combining

In [9]:

```
import numpy as np
import astropy
import ccdproc
from ccdproc import CCDData, Combiner
from astropy import units as u
from astropy.visualization import SqrtStretch
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
from photutils import centroid_com, centroid_1dg, centroid_2dg
from photutils import CircularAperture
from photutils import aperture_photometry
from photutils import Background2D
from photutils import MedianBackground
from scipy.ndimage import shift
import gc
gc.enable()
```

Staring with V band

In [14]:

```
# collecting shifted V band images and reading
images = ccdproc.ImageFileCollection(".",glob_include='proc_NGC_2808_V_*')
scim = [CCDData.read(fn, unit = "adu") for fn in images.files_filtered(PICTTYPE = 1)
```

```
INFO:astropy:using the unit adu passed to the FITS reader instead of t
he unit adu in the FITS file.
INFO:astropy:using the unit adu passed to the FITS reader instead of t
he unit adu in the FITS file.

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

INFO:astropy:using the unit adu passed to the FITS reader instead of t
he unit adu in the FITS file.
INFO:astropy:using the unit adu passed to the FITS reader instead of t
he unit adu in the FITS file.



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

In [16]:

```
# define the positions of some prominent stars in the image
positions = [(859,998), (349,783), (675,787),(570,622), (1085,549), (1237,558)]   #x
apertures = CircularAperture(positions, r=20.0)
```

In [17]:

```
# Complete aperture photometri for each image
phot_table=[]
for idx, thisimage in enumerate(scim):
    phot_table.extend([aperture_photometry(thisimage, apertures)])
    print(idx, phot_table[idx])
```

```
0  id xcenter ycenter    aperture_sum    aperture_sum_err
      pix     pix          adu                adu
--- ------- ------- ------------------ -----------------
  1   859.0   998.0  1431732.213414641               0.0
  2   349.0   783.0 1303607.0049587581               0.0
  3   675.0   787.0 1391505.1374854224               0.0
  4   570.0   622.0 1329839.5757741663               0.0
  5  1085.0   549.0 2049014.4610751367               0.0
  6  1237.0   558.0 1054799.2794533109               0.0
1  id xcenter ycenter    aperture_sum    aperture_sum_err
      pix     pix          adu                adu
--- ------- ------- ------------------ -----------------
  1   859.0   998.0  1433276.040363774               0.0
  2   349.0   783.0 1300709.0711159455               0.0
  3   675.0   787.0 1388567.9142572442               0.0
  4   570.0   622.0  1337878.586269509               0.0
  5  1085.0   549.0  2056182.651859751               0.0
  6  1237.0   558.0 1052773.6935873833               0.0
2  id xcenter ycenter    aperture_sum    aperture_sum_err
```

In [18]:

```
# the reference image for the V band will be the 1stimage
# here we are printing the counts and finding the median values
for idx, thisimage in enumerate(scim):
    print(idx)
    print(phot_table[6]['aperture_sum']/phot_table[idx]['aperture_sum'])
    print(np.ma.median(phot_table[0]['aperture_sum']/phot_table[idx]['aperture_sum']
```

```
0
[0.98757614 0.98005939 0.98357678 0.98330029 0.97372238 0.97454738]
1.0
1
[0.98651239 0.98224293 0.98565733 0.97739186 0.97032783 0.97642246]
1.000423457592539
2
[0.99045887 0.98402532 0.98928751 0.98384922 1.02569767 0.97835184]
1.0039752167869405
3
[0.99595033 0.99147411 0.99195524 0.98893348 0.98290623 0.98751612]
1.0089750266657211
4
[0.99534044 0.99552066 0.99605698 0.99422229 0.98453015 0.98542779]
1.011136030443298
5
[1.00038527 0.99987273 1.00443629 1.00209862 0.98919879 0.99607899]
1.0196670282336453
6
```

In [19]:

```
# scaling the images to our reference images
images = ccdproc.ImageFileCollection(".",glob_include = 'sproc_NGC_2808_V_*') #defir
scim = [CCDData.read(fn) for fn in images.files_filtered()] #reads defined images
for idx, thisimage in enumerate(scim):
    m = np.ma.median(phot_table[6]['aperture_sum'] / phot_table[idx]['aperture_sum']
    print(m)
    scim[idx] = scim[idx].multiply(m * u.adu)

sci_average = ccdproc.combine(scim, method = 'average',dtype = np.float32,
                              minmax_clip = True, minmax_clip_min = -500) #calculati
sci_average.write("NGC_2808_V_average.fits")

sci_median = ccdproc.combine(scim, method = 'median',dtype = np.float32,
                             minmax_clip = True, minmax_clip_min = -500) #calculatir
sci_median.write("NGC_2808_V_median.fits")

del(scim)
collected = gc.collect()
print('Check garbage collection', collected)
```

```
0.9816798402962046
0.9798173943706585
0.9866564141615429
0.9902037944374154
0.9947813614498613
1.0001290040314132
1.0
0.9994458330694227
0.9957011303174386
1.0021473716676543
1.0017728273738
0.9985160500450633
0.9979207990859267
1.0784414315713455
1.0956601054488324
1.096989671127274
1.1042096214187274
1.1053595932685614
1.1214933095634598
1.1619696165347557
1.1728923722270994
1.1843972450897295
1.0245703085142193
1.0137265662521513
1.0202099883420428
1.0400525327951886
1.0414946595259795
1.0566008556327064
1.0513831928330277
1.0511779789692335
Check garbage collection 232
```

R band scale and combine

In [20]:

```
# collecting processed R band images and reading into scim
images = ccdproc.ImageFileCollection(".",glob_include='proc_NGC_2808_R_*')
scim = [CCDData.read(fn, unit = "adu") for fn in images.files_filtered(PICTTYPE = 1)
```

...

In [21]:

```
# define the positions of some prominent stars in the image
positions = [(864,999), (349,784), (675,790),(570,622), (1085,559), (1229,518)]   #x
apertures = CircularAperture(positions, r=20.0)
```

In [22]:

```
# Complete aperture photometry for each image
phot_table=[]
for idx, thisimage in enumerate(scim):
    phot_table.extend([aperture_photometry(thisimage, apertures)])
    print(idx, phot_table[idx])
```

```
0  id xcenter ycenter    aperture_sum    aperture_sum_err
      pix     pix          adu                adu
--- ------- ------- ------------------ ----------------
  1   864.0   999.0  1355392.642351546              0.0
  2   349.0   784.0  1350678.035339181              0.0
  3   675.0   790.0  1403193.311109228              0.0
  4   570.0   622.0 1298024.2413005068              0.0
  5  1085.0   559.0  1818662.568682152              0.0
  6  1229.0   518.0 1164266.0465822308              0.0
1  id xcenter ycenter    aperture_sum    aperture_sum_err
      pix     pix          adu                adu
--- ------- ------- ------------------ ----------------
  1   864.0   999.0 1359954.5659341058              0.0
  2   349.0   784.0 1355206.6734113037              0.0
  3   675.0   790.0    1410175.7263627              0.0
  4   570.0   622.0 1298530.0636995472              0.0
  5  1085.0   559.0 1820630.4071101283              0.0
  6  1229.0   518.0  1163708.919101608              0.0
2  id xcenter ycenter    aperture_sum    aperture_sum_err
```

In [23]:

```
# the reference image for the R band will be the 1st image
# here we are printing the counts and finding the median values
for idx, thisimage in enumerate(scim):
    print(idx)
    print(phot_table[6]['aperture_sum']/phot_table[idx]['aperture_sum'])
    print(np.ma.median(phot_table[0]['aperture_sum']/phot_table[idx]['aperture_sum']
```

...

In [24]:

```
# scaling the images to our reference images
images = ccdproc.ImageFileCollection(".",glob_include = 'sproc_NGC_2808_R_*') #defir
scim = [CCDData.read(fn) for fn in images.files_filtered()] #reads defined images
for idx, thisimage in enumerate(scim):
    m = np.ma.median(phot_table[6]['aperture_sum'] / phot_table[idx]['aperture_sum']
    print(m)
    scim[idx] = scim[idx].multiply(m * u.adu)

sci_average = ccdproc.combine(scim, method = 'average',dtype = np.float32,
                              minmax_clip = True, minmax_clip_min = -500) #calculati
sci_average.write("NGC_2808_R_average.fits")

sci_median = ccdproc.combine(scim, method = 'median',dtype = np.float32,
                             minmax_clip = True, minmax_clip_min = -500) #calculatir
sci_median.write("NGC_2808_R_median.fits")

del(scim)
collected = gc.collect()
print('Check garbage collection', collected)
```

```
0.9792388058340389
0.9780761760225629
0.9761414324416458
0.9897944079634319
0.9860789104576431
0.9938207487208364
1.0
1.0104436441773739
1.0154131351124702
1.0253918407464373
1.0391623220447546
0.9758206149247337
0.9568741927507591
0.9678660695501607
0.960060281739227
0.9557952929721489
0.9661463959556429
0.9653756929570176
0.9706887922772138
0.9728067492840026
0.9677582654442543
Check garbage collection 228
```

I band

In [25]:

```
# collecting processed I band images and reading into scim
images = ccdproc.ImageFileCollection(".",glob_include='proc_NGC_2808_I_*')
scim = [CCDData.read(fn, unit = "adu") for fn in images.files_filtered(PICTTYPE = 1)
```

...

In [26]:

```
# define the positions of some prominent stars in the image
positions = [(347,786), (623,796), (566,639),(1083,562), (1228,515), (1262,376)]   #
apertures = CircularAperture(positions, r=20.0)
```

In [27]:

```
# Complete aperture photometri for each image
phot_table=[]
for idx, thisimage in enumerate(scim):
    phot_table.extend([aperture_photometry(thisimage, apertures)])
    print(idx, phot_table[idx])
```

```
0  id xcenter ycenter    aperture_sum    aperture_sum_err
      pix     pix          adu                adu
--- ------- ------- ------------------ ----------------
  1   347.0   786.0  625917.7902492008              0.0
  2   623.0   796.0 274346.77703536284              0.0
  3   566.0   639.0  483476.2751650661              0.0
  4  1083.0   562.0  672413.0636220854              0.0
  5  1228.0   515.0  459072.7355856062              0.0
  6  1262.0   376.0  638437.9825010813              0.0
1  id xcenter ycenter    aperture_sum   aperture_sum_err
      pix     pix          adu               adu
--- ------- ------- ----------------- ----------------
  1   347.0   786.0  623229.099772279              0.0
  2   623.0   796.0 275761.5499961254              0.0
  3   566.0   639.0 487166.2469345518              0.0
  4  1083.0   562.0 673786.7353796719              0.0
  5  1228.0   515.0 464376.0741237472              0.0
  6  1262.0   376.0 637084.9282684759              0.0
2  id xcenter ycenter    aperture_sum    aperture_sum_err
```

In [28]:

```
# the reference image for the R band will be the 1st image
# here we are printing the counts and finding the median values
for idx, thisimage in enumerate(scim):
    print(idx)
    print(phot_table[0]['aperture_sum']/phot_table[idx]['aperture_sum'])
    print(np.ma.median(phot_table[0]['aperture_sum']/phot_table[idx]['aperture_sum']
```

```
0
[1. 1. 1. 1. 1. 1.]
1.0
1
[1.00431413 0.99486958 0.99242564 0.99796127 0.98857965 1.00212382]
0.9964154224189978
2
[1.00226055 1.0078695  0.99357138 1.00337831 0.99639001 1.00345295]
1.0028194272007924
3
[1.00034496 0.9995126  0.99591073 0.99853352 0.9919414  1.00668054]
0.9990230568481301
4
[1.01922452 1.00645156 1.00420399 1.01831536 1.00914888 1.02499687]
1.0137321214497985
5
[1.01128759 1.00599271 0.99147504 0.99914913 0.99794394 1.00614385]
1.0025709219587466
6
[1.00697453 1.00610073 0.99944903 1.00335458 0.99937255 1.00121692]
1.002285747849737
7
[1.00473531 1.00677948 0.99904823 1.0040686  0.99044227 1.00815044]
1.004401954621645
8
[1.00493034 1.01122312 0.99164172 1.00357324 0.99502795 1.00591953]
1.0042517909016713
9
[1.1146654  1.02074491 1.0756216  1.10990108 1.0731198  1.11164641]
1.092761340946332
10
[1.11133046 1.00953501 1.08558201 1.11028508 1.06113819 1.10031657]
1.0929492909828944
11
[1.09688562 1.01554447 1.07822786 1.09497339 1.06487376 1.09215888]
1.0851933730306216
12
[1.10304022 1.01262731 1.07447647 1.09541732 1.06344812 1.09312253]
1.0837994972378846
13
[1.08678262 1.02548057 1.0615513  1.07843118 1.0589818  1.08097224]
1.0699912440767445
14
[1.09930682 1.02856351 1.06944616 1.09718017 1.06481905 1.09480678]
1.0821264699819184
15
[1.10249984 1.02549404 1.07830112 1.10058418 1.06747772 1.10178864]
1.0894426465286886
16
[1.12132126 1.03513848 1.09617849 1.12250634 1.08805876 1.1234819 ]
```

```
1.1087498742155182
17
[1.10915272 1.02554696 1.07939007 1.11425532 1.07138853 1.10659849]
1.0929942793921514
18
[1.23326291 1.0055213  1.18636468 1.22790999 1.15092021 1.22565982]
1.2060122490413074
19
[1.17196006 1.04388318 1.13621871 1.16784657 1.12568577 1.1649239 ]
1.1505713054066915
20
[1.13647228 1.05910018 1.09968928 1.12480446 1.1051524  1.13461273]
1.1149784339573638
21
[1.15422867 1.05951349 1.12281166 1.15691935 1.10947184 1.14835817]
1.1355849159893572
22
[1.15871515 1.05184555 1.13395289 1.15492407 1.11610165 1.15792792]
1.1444384828760532
23
[1.16870462 1.04367541 1.13088024 1.16768908 1.12464989 1.16183911]
1.1463596724577925
```

In [29]:

```
# scaling the images to our reference images
images = ccdproc.ImageFileCollection(".",glob_include = 'sproc_NGC_2808_I_*') #defir
scim = [CCDData.read(fn) for fn in images.files_filtered()] #reads defined images
for idx, thisimage in enumerate(scim):
    m = np.ma.median(phot_table[6]['aperture_sum'] / phot_table[idx]['aperture_sum']
    print(m)
    scim[idx] = scim[idx].multiply(m * u.adu)

sci_average = ccdproc.combine(scim, method = 'average',dtype = np.float32,
                              minmax_clip = True, minmax_clip_min = -500) #calculati
sci_average.write("NGC_2808_I_average.fits")

sci_median = ccdproc.combine(scim, method = 'median',dtype = np.float32,
                             minmax_clip = True, minmax_clip_min = -500) #calculatir
sci_median.write("NGC_2808_I_median.fits")

del(scim)
collected = gc.collect()
print('Check garbage collection', collected)
```

```
0.9977205994763121
0.9937987290655439
0.9985196179446082
0.9943234364559294
1.0109738079577113
0.9992315674081478
1.0
1.0001368046677932
0.9990939514193924
1.0912024202412547
1.0925798311461625
1.0840553014977417
1.083411869759026
1.0684810527147683
1.0808642471333088
1.0868796165803447
1.1051687591794577
1.0907277956743573
1.2054116640994694
1.1501765396576726
1.113445053655943
1.1348324279197568
1.1426338179955549
1.1459653099263574
Check garbage collection 300
```

In [ ]: