

Lab 4 Radial Profiles of Galaxies

Patrick Selep

Abstract

Several galaxies were observed in both red and blue filters. Examples of both spiral and elliptical galaxies were analyzed and their radial profiles plotted. Curves were fit using both exponential and de Vaucouleurs models.

These same images were also analyzed using differential and aperture photometry and the total luminosity of the galaxies were calculated in terms of the luminosity of our own sun.

Observations and Image Processing

Radial profiles were constructed from observations taken on 03/06/2020, generally a series of five two-minute exposures in each filter using 2x2 binning. The observations were made with the UWM 14" telescope and CCD camera.

AstroImageJ (AIJ) was used extensively to calibrate the images with bias, dark and flat field images. Astrometry.net was used to obtain astrometric solutions for each image. AIJ was then used to stack and sum the images. AIJ's Radial Profile functionality was used to produce a data table for subsequent analysis. (Collins, 2017)

The output file was then read into Python and the data plotted in a radial profile.

The same stacked images in each filter were subsequently aligned, stacked and a difference image produced subtracting blue from red. The resulting image was also processed to produce a radial profile across filters.

The stacked images in each filter were also analyzed using DS9 to calculate the total luminosity of the galaxy.

Spiral and Elliptical Galaxies

An example of each type of galaxy was analyzed; M81 as the spiral galaxy, M87 as the elliptical.

Curves were fit to the radial profile using exponential and de Vaucouleurs models. The exponential curve is preferred for the spiral galaxy and the de Vaucouleurs is recommended for the elliptical.

The R-B radial profile highlighted a distinct difference between the red and blue profiles in the different galaxy types. In the spiral galaxy the red was concentrated at the center or conversely the blue was more prevalent further out. This follows from the prevalence of older stars at the center and star forming regions further out. In the elliptical galaxy the concentration was less pronounced as there are typically fewer blue stars present relative to red overall as the stars are older and the gas and dust have been used up.

The total luminosity was calculated based on differential and aperture photometry comparing the collected light of the galaxy with that of a star of known magnitude. The total luminosity calculations put the number on the order tens of billions of suns which is within a couple orders of magnitude but low. This might be attributed to the typical light pollution and nearly full moon on the night the observations were taken. These factors would have the effect of washing out light that otherwise might have been attributed to the galaxy. Also, M87 was the only elliptical galaxy visible that night and was no where near as prominent as the spiral galaxies observed.

de Vaucouleurs' law

Radial profiles of elliptical galaxies are typically fit using the de Vaucouleurs' (1948) law

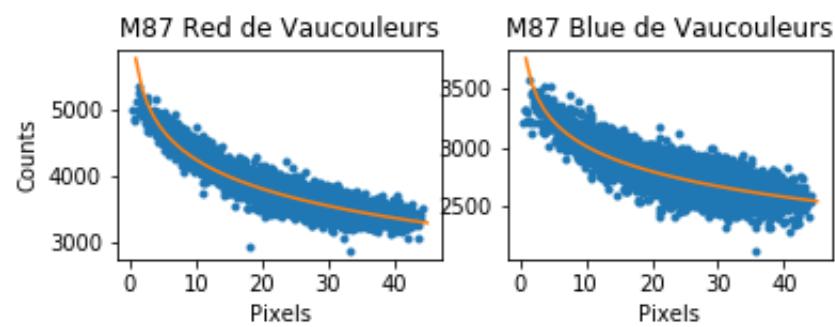
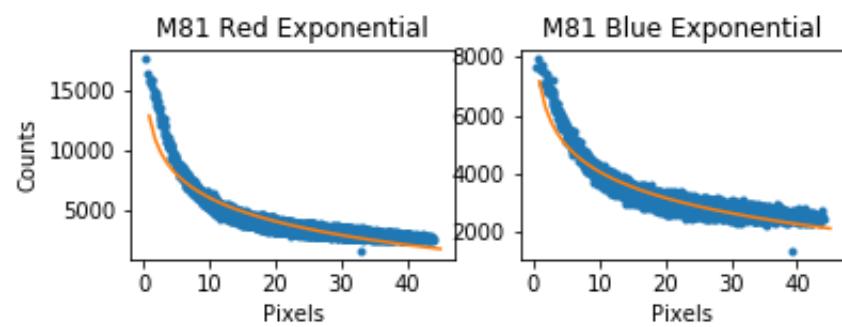
$$I(R) = I_0 \exp(-kR^{1/4})$$

Absolute Magnitude

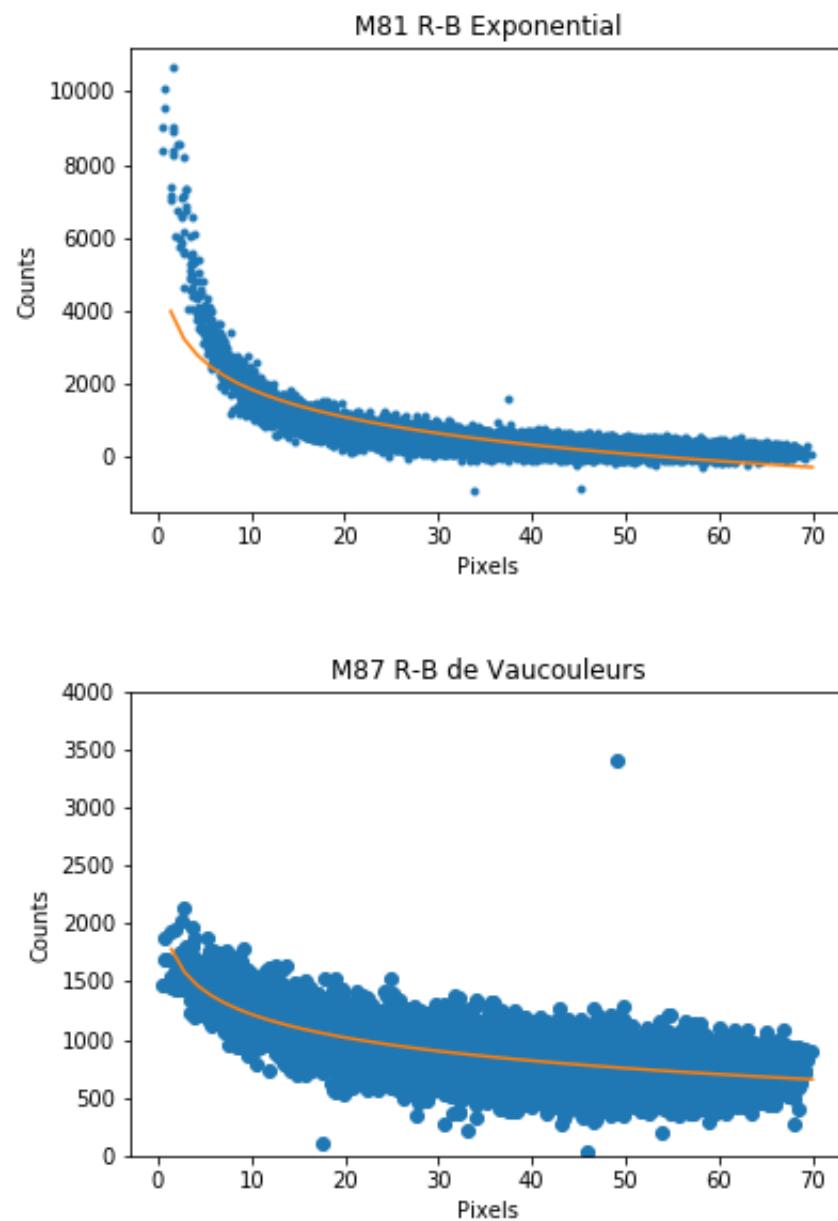
Absolute Magnitude, M , was calculated as follow:

$$M = m - 2.5 \log_{10}(d/10)^2$$

Individual filters

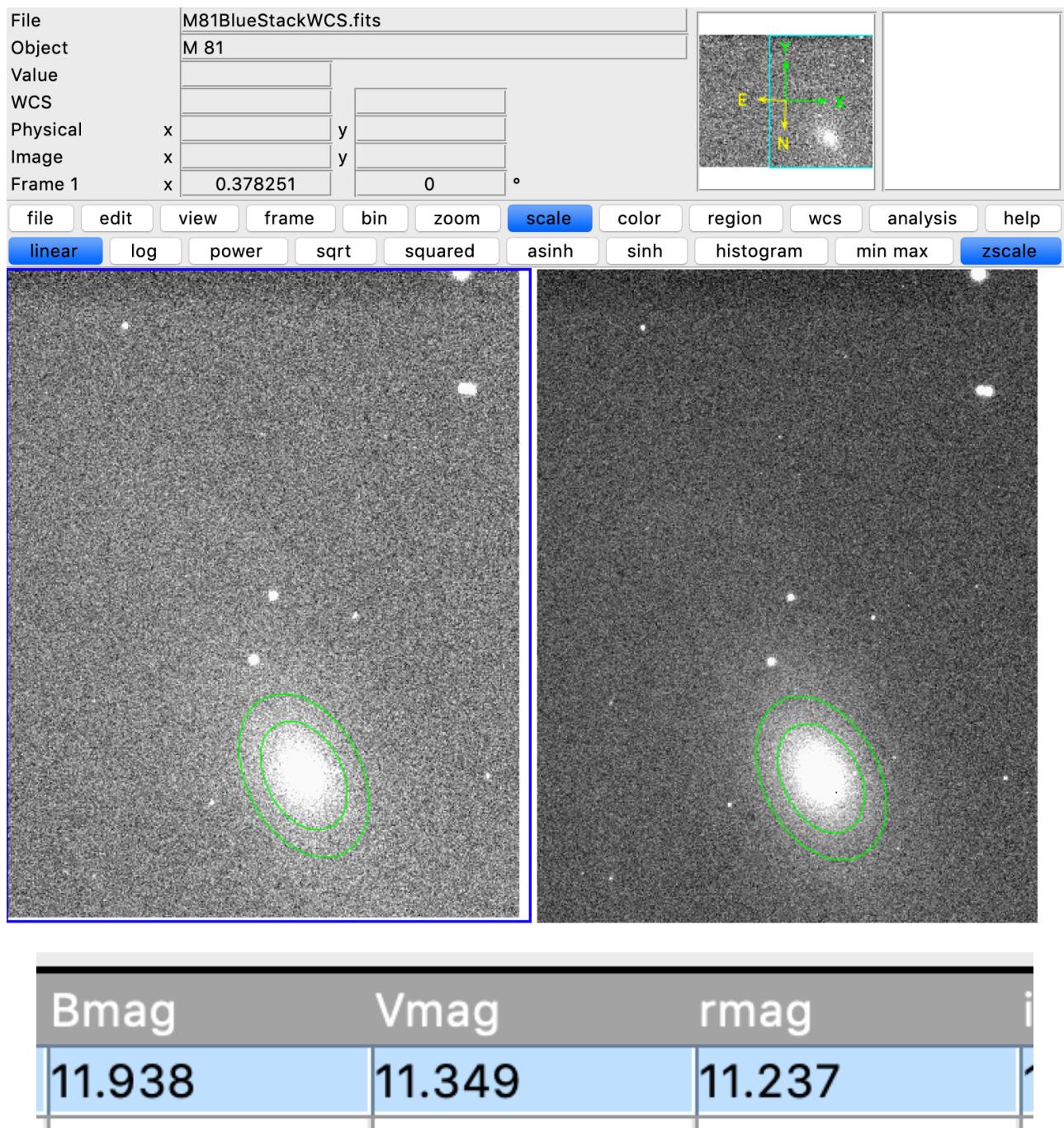


R-B Filters

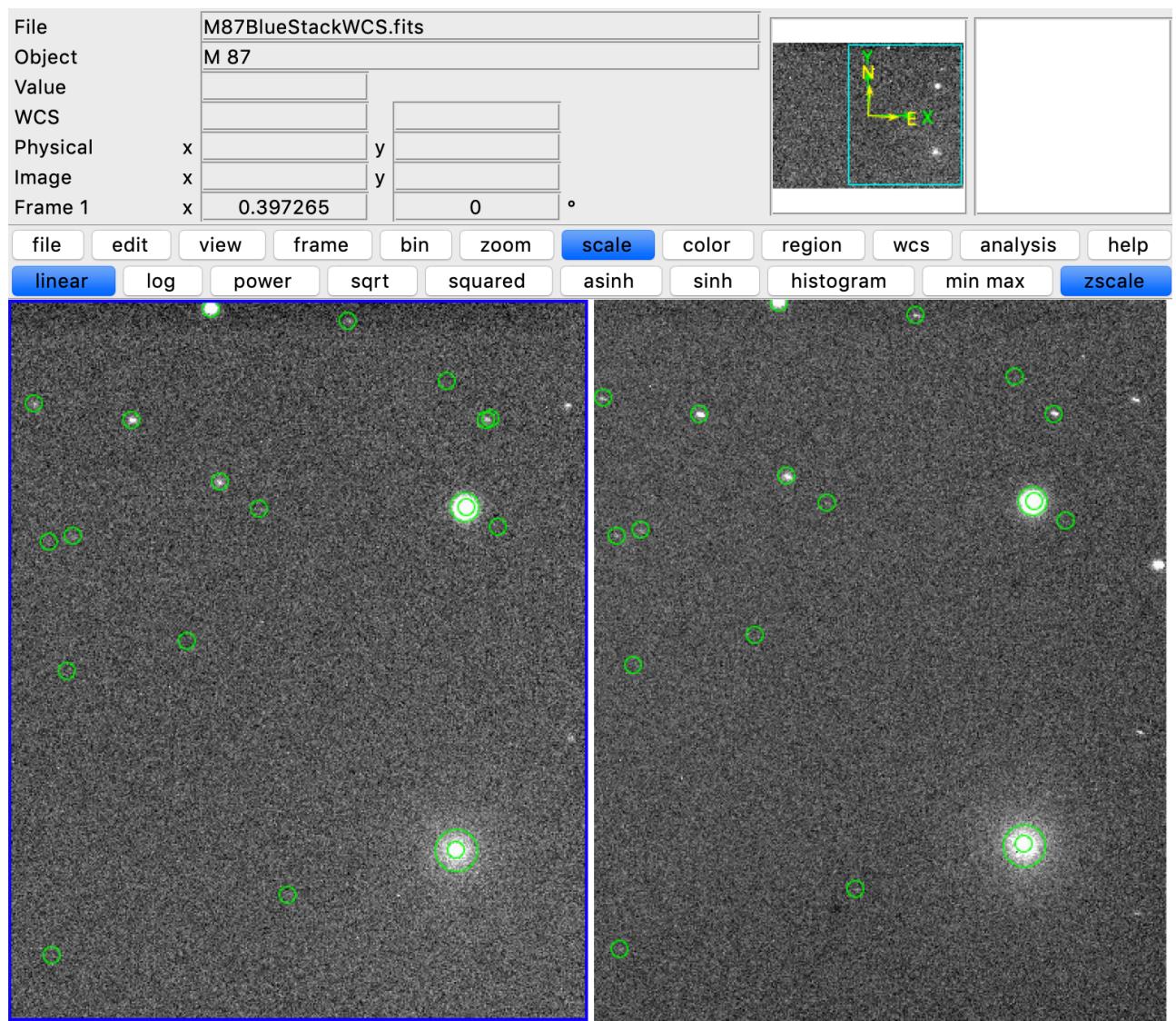


Total Luminosity

M81



M87



Bmag	Vmag	rmag
9.693	9.297	9.005

In [28]:

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy as scipy
4 from astropy.stats import sigma_clipped_stats
5 from scipy import stats
6
7 Msunb = 5.44    # Wilmer, C. (2018)
8 mRefb = 11.938

```

```
9 f81b = 69406464
10 fRefb = 2026251
11 m81b = mRefb - 2.5 * np.log10(f81b/fRefb)
12 DistM81 = 3.6 * 10**6
13 M81b = 5 + m81b - 5 * np.log10(DistM81)
14 Suns81b = 100**((Msunb-M81b)/5)
15
16 print("Apparent magnitude of M81 in Blue: %0.3f" % (m81b))
17 print("Absolute magnitude of M81 in Blue: %0.3f" % (M81b))
18 print("In terms of suns: %0.2f billion" % (Suns81b/10**9))
19 print()
20
21 Msunr = 4.43 # Wilmer, C. (2018)
22 mRefr = 11.237
23 f81r = 76651327
24 fRefr = 1526263
25 m81r = mRefr - 2.5 * np.log10(f81r/fRefr)
26 M81r = 5 + m81r - 5 * np.log10(DistM81)
27 Suns81r = 100**((Msunr-M81r)/5)
28
29 print("Apparent magnitude of M81 in Red: %0.3f" % (m81r))
30 print("Absolute magnitude of M81 in Red: %0.3f" % (M81r))
31 print("In terms of suns: %0.2f billion" % (Suns81r/10**9))
32 print()
33
34 mRefb = 9.693
35 fRefb = 12774613
36 f87b = 11496322
37 m87b = mRefb - 2.5 * np.log10(f87b/fRefb)
38 DistM87 = 16.4 * 10**6
39 M87b = 5 + m87b - 5 * np.log10(DistM87)
40 Suns87b = 100**((Msunb-M87b)/5)
41
42 print("Apparent magnitude of M87 in Blue: %0.3f" % (m87b))
43 print("Absolute magnitude of M87 in Blue: %0.3f" % (M87b))
44 print("In terms of suns: %0.2f billion" % (Suns87b/10**9))
45 print()
46
47 mRefr = 9.005
48 f87r = 15572807
49 fRefr = 16062810
50 m87r = mRefr - 2.5 * np.log10(f87r/fRefr)
51 M87r = 5 + m87r - 5 * np.log10(DistM87)
52 Suns87r = 100**((Msunr-M87r)/5)
53
54 print("Apparent magnitude of M87 in Red: %0.3f" % (m87r))
55 print("Absolute magnitude of M87 in Red: %0.3f" % (M87r))
56 print("In terms of suns: %0.2f billion" % (Suns87r/10**9))
57 print()
```

Apparent magnitude of M81 in Blue: 8.101

```
    ```
 Absolute magnitude of M81 in Blue: -19.680
 In terms of suns: 11.17 billion

 Apparent magnitude of M81 in Red: 6.985
 Absolute magnitude of M81 in Red: -20.797
 In terms of suns: 12.32 billion

 Apparent magnitude of M87 in Blue: 9.807
 Absolute magnitude of M87 in Blue: -21.267
 In terms of suns: 48.16 billion

 Apparent magnitude of M87 in Red: 9.039
 Absolute magnitude of M87 in Red: -22.036
 In terms of suns: 38.57 billion
```

## Full AIJ Profile

# M81 Analysis - Spiral Galaxy Radial Profile Curve Fitting - Exponential Curve

In [3]:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy as scipy
4 from astropy.stats import sigma_clipped_stats
5 from scipy import stats
6 import warnings
7 warnings.filterwarnings('ignore')

8
9
10 def exponential(x, a, k, b):
11 return a*np.log10(x**k) + b
12
13 #def SecondOrder(x, a, k, b):
14 # return a*np.log10(k*x**.25) + b
15
16 fig = plt.figure()
17
18 data = np.genfromtxt("M81RedPlotValues.tsv", delimiter="\t", skip_he
19
20 #x, y = xy.T
21 x = (data[:,0])
22 y = (data[:,1])
23
24
25 mean, median, std = sigma_clipped_stats(y)
```

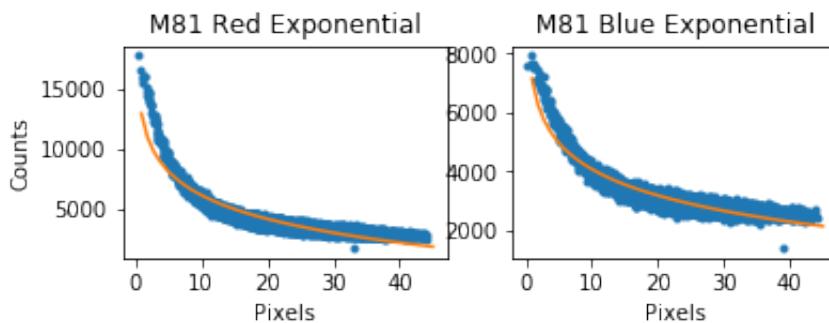
```
26 #print(min(y),max(y),mean,median,std)
27 #print(x,y)
28
29 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(expone
30 perr_exponential = np.sqrt(np.diag(pcov_exponential))
31 print ("Exponential on the raw data")
32 #print (popt_exponential, pcov_exponential)
33 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expo
34 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
35 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
36
37 xtest = np.linspace(0,45)
38 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +
39
40 plt.subplot(2, 2, 1)
41 plt.plot(x, y, linestyle = 'None', marker = '.')
42 plt.plot(xtest,testy)
plt.ylim(10000,70000)
44 plt.title('M81 Red Exponential')
45 plt.xlabel('Pixels')
46 plt.ylabel('Counts')
plt.show()
48
49 data = np.genfromtxt("M81BluePlotValues.tsv", delimiter="\t", skip_h
50
51 #x, y = xy.T
52 x = (data[:,0])
53 y = (data[:,1])
54
55 #x, y = xy.T
56 mean, median, std = sigma_clipped_stats(y)
print(min(y),max(y),mean,median,std)
58 #print(x,y)
59
60 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(expone
61 perr_exponential = np.sqrt(np.diag(pcov_exponential))
62 print ("Exponential on the raw data")
63 #print (popt_exponential, pcov_exponential)
64 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expo
65 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
66 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
67
68 xtest = np.linspace(0,45)
69 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +
70
71 plt.subplot(2, 2, 2)
72 plt.plot(x, y, linestyle = 'None', marker = '.')
73 plt.plot(xtest,testy)
plt.ylim(10000,70000)
75 plt.title('M81 Blue Exponential')
plt.xlabel('Pixels')
```

```

 , xlabel='Pixels',
77 #plt.ylabel('Counts')
78 plt.savefig("M81exponential.png")
79 plt.show()

```

Exponential on the raw data  
pre-exponential factor = -6527.2206979377 (+/-) 39.8917972998  
rate constant = 2.3298 (+/-) 393289.5814  
yintercept = 15062.6515 (+/-) 477989226.6355  
Exponential on the raw data  
pre-exponential factor = -2967.4865363590 (+/-) 16.9161690243  
rate constant = 14.0544 (+/-) 1025358.7545  
yintercept = 10447.1207 (+/-) 93914381.9431



## M87 Analysis - Spiral Galaxy Radial Profile Curve Fitting - de Vaucouleurs Curve

In [4]:

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy as scipy
4 from astropy.stats import sigma_clipped_stats
5 from scipy import stats
6 import warnings
7 warnings.filterwarnings('ignore')

8

9
10 #def exponential(x, a, k, b):
11 # return a*np.log10(x*k) + b
12
13 def SecondOrder(x, a, k, b):
14 return a*np.log10(k*x**.25) + b
15
16 fig = plt.figure()
17
18 data = np.genfromtxt("M87RedPlotValues.tsv", delimiter="\t", skip_he
19
20 #x, y = xy.T
21 x = (data[:,0])
22 v = (data[:,1])

```

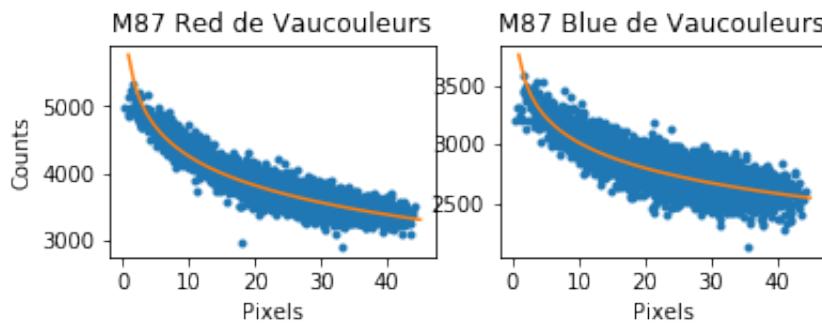
```
--
23
24
25 mean, median, std = sigma_clipped_stats(y)
26 #print(min(y),max(y),mean,median,std)
27 #print(x,y)
28
29 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
30 perr_exponential = np.sqrt(np.diag(pcov_exponential)))
31 print ("de Vaucouleurs on the raw data")
32 #print (popt_exponential, pcov_exponential)
33 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
34 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
35 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
36
37 xtest = np.linspace(0,45)
38 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.
39
40 plt.subplot(2, 2, 1)
41 plt.plot(x, y, linestyle = 'None', marker = '.')
42 plt.plot(xtest,testy)
43 #plt.ylim(10000,70000)
44 plt.title('M87 Red de Vaucouleurs')
45 plt.xlabel('Pixels')
46 plt.ylabel('Counts')
47 #plt.show()
48
49 data = np.genfromtxt("M87BluePlotValues.tsv", delimiter="\t", skip_h
50
51 #x, y = xy.T
52 x = (data[:,0])
53 y = (data[:,1])
54
55 #x, y = xy.T
56 mean, median, std = sigma_clipped_stats(y)
57 #print(min(y),max(y),mean,median,std)
58 #print(x,y)
59
60 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
61 perr_exponential = np.sqrt(np.diag(pcov_exponential)))
62 print ("de Vaucouleurs on the raw data")
63 #print (popt_exponential, pcov_exponential)
64 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
65 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
66 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
67
68 xtest = np.linspace(0,45)
69 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.
70
71 plt.subplot(2, 2, 2)
72 plt.plot(x, y, linestyle = 'None', marker = '.')
```

```

73 plt.plot(xtest,testy)
74 #plt.ylim(10000,70000)
75 plt.title('M87 Blue de Vaucouleurs')
76 plt.xlabel('Pixels')
77 #plt.ylabel('Counts')
78 plt.savefig("M87deVaucouleurs.png")
79 plt.show()

```

de Vaucouleurs on the raw data  
pre-exponential factor = -5838.8799644059 (+/-) 38.7672592974  
rate constant = 30.2440 (+/-) 1383994.1081  
yintercept = 14359.8604 (+/-) 116040309.3519  
de Vaucouleurs on the raw data  
pre-exponential factor = -2859.3860296192 (+/-) 29.6808886044  
rate constant = 45.3364 (+/-) 1777622.6717  
yintercept = 8466.4980 (+/-) 48691125.9814



## R-B Filter Analysis - Galaxy Radial Profile Curve Fitting - Exponential Curve

In [8]:

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy as scipy
4 from astropy.stats import sigma_clipped_stats
5 from scipy import stats
6
7 def exponential(x, a, k, b):
8 return a*np.log10(x*k) + b
9
10 #def SecondOrder(x, a, k, b):
11 # return a*np.log10(k*x**.25) + b
12
13 fig = plt.figure()
14
15 data = np.genfromtxt("M81R_BStackWCSPlotValues.tsv", delimiter="\t",
16
17 #x, y = xy.T
18 x = (data[:,0])

```

```
19 y = (data[:,1])
20
21
22 mean, median, std = sigma_clipped_stats(y)
23 #print(min(y),max(y),mean,median,std)
24 #print(x,y)
25
26 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(expone
27 perr_exponential = np.sqrt(np.diag(pcov_exponential))
28 print ("Exponential on the raw data")
29 #print (popt_exponential, pcov_exponential)
30 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
31 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
32 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
33
34 xtest = np.linspace(0,70)
35 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +
36
37 plt.plot(x, y, linestyle = 'None', marker = '.')
38 plt.plot(xtest,testy)
39 #plt.ylim(0,4000)
40 plt.title('M81 R-B Exponential')
41 plt.xlabel('Pixels')
42 plt.ylabel('Counts')
43 plt.savefig("M81R_BExponential.png")
44 plt.show()
45
46 plt.subplot(2, 2, 1)
47 plt.plot(x, y, linestyle = 'None', marker = '.')
48 plt.plot(xtest,testy)
49 #plt.ylim(10000,70000)
50 plt.title('M81 R-B Exponential')
51 plt.xlabel('Pixels')
52 plt.ylabel('Counts')
53 #plt.show()
54
55 data = np.genfromtxt("M87R_BStackWCSPlotValues.tsv", delimiter="\t",
56
57 #x, y = xy.T
58 x = (data[:,0])
59 y = (data[:,1])
60
61 #x, y = xy.T
62 mean, median, std = sigma_clipped_stats(y)
63 #print(min(y),max(y),mean,median,std)
64 #print(x,y)
65
66 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(expone
67 perr_exponential = np.sqrt(np.diag(pcov_exponential))
68 print ("Exponential on the raw data")
69 #print (popt_exponential, pcov_exponential)
```

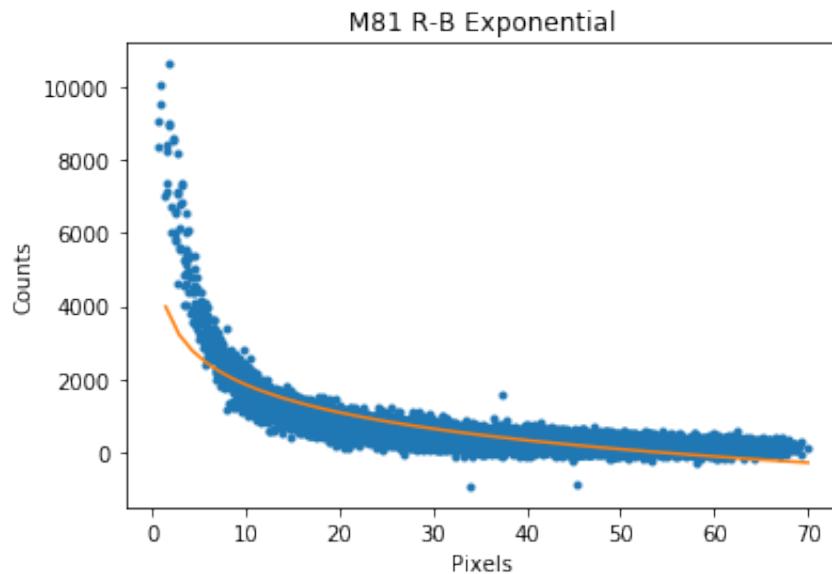
```

70 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponentia
71 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], perr_
72 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
73
74 xtest = np.linspace(0,70)
75 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +
76
77 plt.subplot(2, 2, 2)
78 plt.plot(x, y, linestyle = 'None', marker = '.')
79 plt.plot(xtest,testy)
80 plt.ylim(0,10000)
81 plt.title('M87 R-B Exponential')
82 plt.xlabel('Pixels')
83 #plt.ylabel('Counts')
84 plt.show()
85
86

```

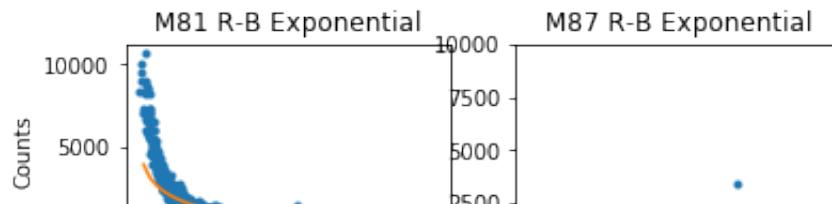
Exponential on the raw data

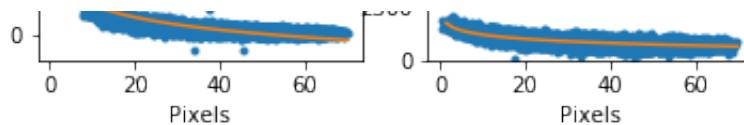
pre-exponential factor = -2525.9658709418 (+/-) 17.0883725707  
 rate constant = 0.2186 (+/-) 55406.0756  
 yintercept = 2706.0569 (+/-) 278461450.3703



Exponential on the raw data

pre-exponential factor = -662.9317062439 (+/-) 7.3144175569  
 rate constant = 16.9436 (+/-) 2218230.4545  
 yintercept = 2696.8668 (+/-) 37692347.2903





## R-B Filter Analysis - Galaxy Radial Profile Curve Fitting - de Vaucouleurs Curve

In [9]:

```

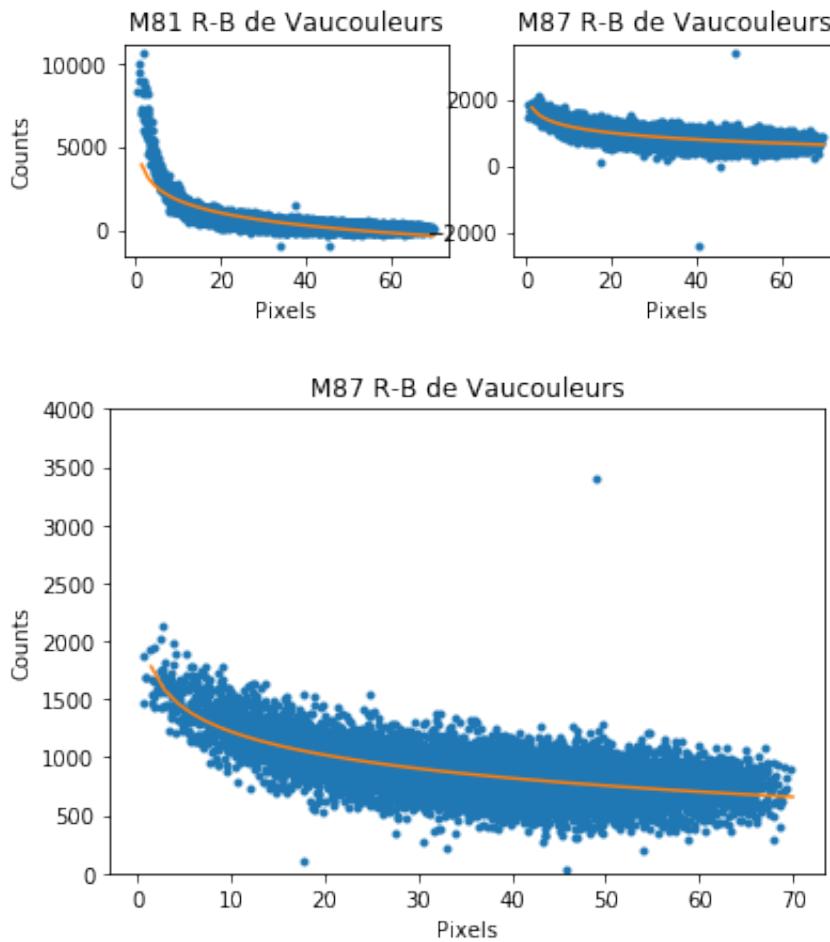
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy as scipy
4 from astropy.stats import sigma_clipped_stats
5 from scipy import stats
6
7 #def exponential(x, a, k, b):
8 # return a*np.log10(x*k) + b
9
10 def SecondOrder(x, a, k, b):
11 return a*np.log10(k*x**.25) + b
12
13 fig = plt.figure()
14
15 data = np.genfromtxt("M81R_BStackWCSPlotValues.tsv", delimiter="\t",
16
17 #x, y = xy.T
18 x = (data[:,0])
19 y = (data[:,1])
20
21
22 mean, median, std = sigma_clipped_stats(y)
23 #print(min(y),max(y),mean,median,std)
24 #print(x,y)
25
26 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
27 perr_exponential = np.sqrt(np.diag(pcov_exponential))
28 print ("de Vaucouleurs on the raw data")
29 #print (popt_exponential, pcov_exponential)
30 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponen
31 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
32 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
33
34 xtest = np.linspace(0,70)
35 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.
36
37 plt.subplot(2, 2, 1)
38 plt.plot(x, y, linestyle = 'None', marker = '.')
39 plt.plot(xtest,testy)
40 #plt.ylim(10000,70000)
41

```

```
41 plt.title('M87 R-B de Vaucouleurs')
42 plt.xlabel('Pixels')
43 plt.ylabel('Counts')
44 #plt.show()
45
46 data = np.genfromtxt("M87R_BStackWCSPlotValues.tsv", delimiter="\t",
47
48 #x, y = xy.T
49 x = (data[:,0])
50 y = (data[:,1])
51
52 #x, y = xy.T
53 mean, median, std = sigma_clipped_stats(y)
print(min(y), max(y), mean, median, std)
print(x,y)
54
55 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
56 perr_exponential = np.sqrt(np.diag(pcov_exponential))
57 print ("de Vaucouleurs on the raw data")
58 #print (popt_exponential, pcov_exponential)
59 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponent
60 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
61 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
62
63 xtest = np.linspace(0,70)
64 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.
65
66 plt.subplot(2, 2, 2)
67 plt.plot(x, y, linestyle = 'None', marker = '.')
68 plt.plot(xtest,testy)
plt.ylim(10000,70000)
69 plt.title('M87 R-B de Vaucouleurs')
70 plt.xlabel('Pixels')
71 plt.ylabel('Counts')
72 plt.show()
73
74 plt.plot(x, y, linestyle = 'None', marker = '.')
75 plt.plot(xtest,testy)
76 plt.ylim(0,4000)
77 plt.title('M87 R-B de Vaucouleurs')
78 plt.xlabel('Pixels')
79 plt.ylabel('Counts')
80 plt.savefig("M87R_BdeVaucouleurs.png")
81 plt.show()
```

```
de Vaucouleurs on the raw data
pre-exponential factor = -10103.8634926685 (+/-) inf
rate constant = 1.0211 (+/-) inf
yintercept = 4465.7305 (+/-) inf
de Vaucouleurs on the raw data
pre-exponential factor = -2651.7268464935 (+/-) 29.2578685808
```

```
rate constant = 4.6539 (+/-) 278097.9873
yintercept = 3652.9914 (+/-) 68816496.1883
```



In [ ]:

1

## Data supporting Total Luminosity - DS9 Differential Aperture Photometry

### M81 Blue

Comparing the brightness of the galaxy to the brightest star in the Blue filter:

In [ ]:

```
1 Ellipse
2
3 center=148.88831 69.066062
4 fk5
5 1 pixel = 0.59893708 arcsec
6
7 reg sum error area surf_bri surf_err
8 (arcsec**2) (sum/arcsec**2) (sum/arcsec**2)
9 -----
10 1 69406464 8331.05 10031.8 6918.67 0.830468
11
12 reg sum npix mean median min max var stddev rms
13 -----
14 1 69406464 27965 2481.9 2401 1375 7930 139577 373.0
15
16 Circle Mag 11.338
17
18 {148.937561500 +69.029386700 796-020040 148.9375615 +69.0293867 19 1
19
20 center=148.93874 69.029964
21 fk5
22 1 pixel = 0.59893708 arcsec
23
24 reg sum error area surf_bri surf_err
25 (arcsec**2) (sum/arcsec**2) (sum/arcsec**2)
26 -----
27 1 2026251 1423.46 182.591 11097.2 7.7959
28
29 reg sum npix mean median min max var stddev rms
30 -----
31 1 2026251 509 3980.85 2607 2050 29016 1.39178e+07 3730.65
```

## M81 Red

Comparing the brightness of the galaxy to the brightest star in the Red filter:

In [ ]:

```
1 Ellipse
2
3 center=148.88831 69.066062
4 fk5
5 1 pixel = 0.59830185 arcsec
6
7 reg sum error area surf_bri surf_err
8 (arcsec**2) (sum/arcsec**2) (sum/arcsec**2)
9 -----
10 1 76651327 8755.07 10032.7 7640.16 0.872655
11
12 reg sum npix mean median min max var stddev rms
13 -----
14 1 76651327 28027 2734.91 2543 0 22279 619839 787.299
15
16 Circle 11.338
17
18 {148.937561500 +69.029386700 796-020040 148.9375615 +69.0293867 19 1
19
20 center=148.93729 69.029608
21 fk5
22 1 pixel = 0.59830185 arcsec
23
24 reg sum error area surf_bri surf_err
25 (arcsec**2) (sum/arcsec**2) (sum/arcsec**2)
26 -----
27 1 1526263 1235.42 94.8608 16089.5 13.0235
28
29 reg sum npix mean median min max var stddev rms
30 -----
31 1 1526263 265 5759.48 2898 2217 49463 5.55042e+07 7450.11
```

## M87 Blue

Comparing the brightness of the galaxy to the brightest star in the Blue filter:

In [ ]:

```
1 Galaxy
2
3
4 center=187.70621 12.390445
5 fk5
6 1 pixel = 0.59883966 arcsec
7
8 reg sum error area surf_bri surf_err
9 (arcsec**2) (sum/arcsec**2) (sum/arcsec**2)
10 -----
11 1 11496322 3390.62 1489.66 7717.41 2.2761
12
13 reg sum npix mean median min max var stddev rms
14 -----
15 1 11496322 4154 2767.53 2737 2128 3580 35543.4 188.9
16
17 Star 8.860
18
19 {187.703899200 +12.486724800 513-054652 187.7038992 +12.4867248 9 8.8
20
21 center=187.70373 12.486656
22 fk5
23 1 pixel = 0.59883966 arcsec
24
25 reg sum error area surf_bri surf_err
26 (arcsec**2) (sum/arcsec**2) (sum/arcsec**2)
27 -----
28 1 12774613 3574.16 654.82 19508.6 5.45823
29
30 reg sum npix mean median min max var stddev rms
31 -----
32 1 12774613 1826 6995.95 3290 2435 64144 1.12761e+08
```

## M87 Red

Comparing the brightness of the galaxy to the brightest star in the Red filter:

In [ ]:

```

1 Galaxy
2
3 center=187.70621 12.390445
4 fk5
5 1 pixel = 0.59882223 arcsec
6
7 reg sum error area surf_bri surf_err
8 (arcsec**2) (sum/arcsec**2) (sum/arcsec**2)
9 -----
10 1 15572807 3946.24 1489.93 10452 2.6486
11
12 reg sum npix mean median min max var stddev rms
13 -----
14 1 15572807 4155 3747.97 3655 2884 5342 122046 349.1
15
16
17 Star 8.860
18
19 {187.703899200 +12.486724800 513-054652 187.7038992 +12.4867248 9 8.860
20
21 center=187.70373 12.486656
22 fk5
23 1 pixel = 0.59882223 arcsec
24
25 reg sum error area surf_bri surf_err
26 (arcsec**2) (sum/arcsec**2) (sum/arcsec**2)
27 -----
28 1 16062810 4007.84 654.065 24558.4 6.1276
29
30 reg sum npix mean median min max var stddev rms
31 -----
32 1 16062810 1824 8806.37 4278 3114 65273 1.63528e+08
33
34
35
36

```

## Supporting Analysis

### M87 Analysis - Elliptical Galaxy Radial Profile Curve Fitting - Exponential Curve

In [2]:

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy as scipy

```

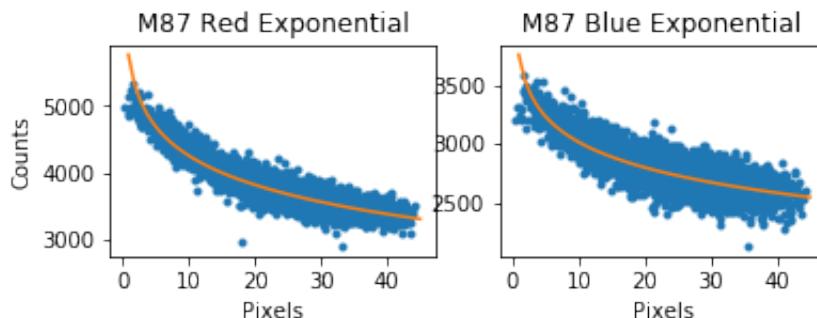
```
4 from astropy.stats import sigma_clipped_stats
5 from scipy import stats
6 import warnings
7 warnings.filterwarnings('ignore')
8
9
10 def exponential(x, a, k, b):
11 return a*np.log10(x*k) + b
12
13 #def SecondOrder(x, a, k, b):
14 # return a*np.log10(k*x**.25) + b
15
16 fig = plt.figure()
17
18 data = np.genfromtxt("M87RedPlotValues.tsv", delimiter="\t", skip_he
19
20 #x, y = xy.T
21 x = (data[:,0])
22 y = (data[:,1])
23
24
25 mean, median, std = sigma_clipped_stats(y)
26 #print(min(y),max(y),mean,median,std)
27 #print(x,y)
28
29 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(expone
30 perr_exponential = np.sqrt(np.diag(pcov_exponential))
31 print ("Exponential on the raw data")
32 #print (popt_exponential, pcov_exponential)
33 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponen
34 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
35 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
36
37 xtest = np.linspace(0,45)
38 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +
39
40 plt.subplot(2, 2, 1)
41 plt.plot(x, y, linestyle = 'None', marker = '.')
42 plt.plot(xtest,testy)
plt.ylim(10000,70000)
44 plt.title('M87 Red Exponential')
45 plt.xlabel('Pixels')
46 plt.ylabel('Counts')
plt.show()
48
49 data = np.genfromtxt("M87BluePlotValues.tsv", delimiter="\t", skip_h
50
51 #x, y = xy.T
52 x = (data[:,0])
53 y = (data[:,1])
54
```

```

55 #x, y = xy.T
56 mean, median, std = sigma_clipped_stats(y)
57 #print(min(y),max(y),mean,median,std)
58 #print(x,y)
59
60 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(expone
61 perr_exponential = np.sqrt(np.diag(pcov_exponential))
62 print ("Exponential on the raw data")
63 #print (popt_exponential, pcov_exponential)
64 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
65 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
66 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
67
68 xtest = np.linspace(0,45)
69 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +
70
71 plt.subplot(2, 2, 2)
72 plt.plot(x, y, linestyle = 'None', marker = '.')
73 plt.plot(xtest,testy)
74 #plt.ylim(10000,70000)
75 plt.title('M87 Blue Exponential')
76 plt.xlabel('Pixels')
77 #plt.ylabel('Counts')
78 plt.savefig("M87exponential.png")
79 plt.show()

```

Exponential on the raw data  
pre-exponential factor = -1459.7199939749 (+/-) 9.6901457507  
rate constant = 10610.8435 (+/-) 496067472.4748  
yintercept = 11591.0500 (+/-) 29611851.6830  
Exponential on the raw data  
pre-exponential factor = -714.8465042582 (+/-) 7.4201598756  
rate constant = 986.1767 (+/-) 68607035.7887  
yintercept = 5870.2950 (+/-) 21597873.6918



## M81 Analysis - Spiral Galaxy Radial Profile Curve Fitting - de Vaucouleurs Curve

In [5]:

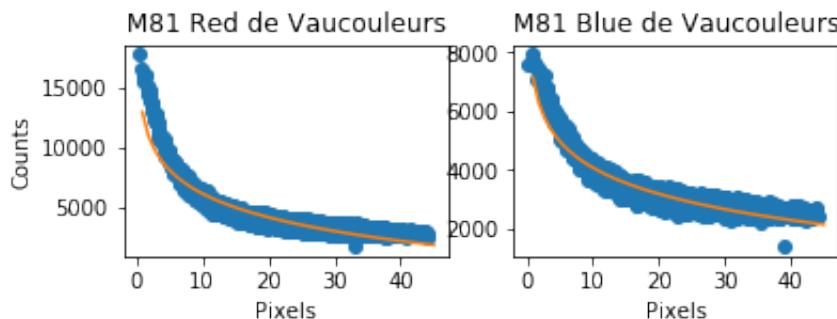
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy as scipy
4 from astropy.stats import sigma_clipped_stats
5 from scipy import stats
6
7 #def exponential(x, a, k, b):
8 # return a*np.log10(x*k) + b
9
10 def SecondOrder(x, a, k, b):
11 return a*np.log10(k*x**.25) + b
12
13 fig = plt.figure()
14
15 data = np.genfromtxt("M81RedPlotValues.tsv", delimiter="\t", skip_he
16
17 #x, y = xy.T
18 x = (data[:,0])
19 y = (data[:,1])
20
21
22 mean, median, std = sigma_clipped_stats(y)
print(min(y), max(y), mean, median, std)
print(x, y)
23
24
25 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
26 perr_exponential = np.sqrt(np.diag(pcov_exponential))
27 print ("de Vaucouleurs on the raw data")
28 #print (popt_exponential, pcov_exponential)
29 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expon
30 print ("rate constant = %0.4f (+/-) %0.4f" % (popt_exponential[1], pe
31 print ("yintercept = %0.4f (+/-) %0.4f" % (popt_exponential[2], perr_
32
33 xtest = np.linspace(0,45)
34 testy = (popt_exponential[0]*np.log10((popt_exponential[1])**xtest**.
35
36
37 plt.subplot(2, 2, 1)
38 plt.plot(x, y, linestyle = 'None', marker = 'o')
39 plt.plot(xtest,testy)
40 #plt.ylim(10000,70000)
41 plt.title('M81 Red de Vaucouleurs')
42 plt.xlabel('Pixels')
43 plt.ylabel('Counts')
44 #plt.show()
45
46 data = np.genfromtxt("M81BluePlotValues.tsv", delimiter="\t", skip_h
47
48 #x, y = xy.T
49 x = (data[:,0])
50 y = (data[:,1])
```

```

51
52 #x, y = xy.T
53 mean, median, std = sigma_clipped_stats(y)
54 #print(min(y),max(y),mean,median,std)
55 #print(x,y)
56
57 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
58 perr_exponential = np.sqrt(np.diag(pcov_exponential))
59 print ("de Vaucouleurs on the raw data")
60 #print (popt_exponential, pcov_exponential)
61 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponentia
62 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
63 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
64
65 xtest = np.linspace(0,45)
66 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.
67
68 plt.subplot(2, 2, 2)
69 plt.plot(x, y, linestyle = 'None', marker = 'o')
70 plt.plot(xtest,testy)
71 #plt.ylim(10000,70000)
72 plt.title('M81 Blue de Vaucouleurs')
73 plt.xlabel('Pixels')
74 #plt.ylabel('Counts')
75 plt.savefig("M81deVaucouleurs.png")
76 plt.show()

```

de Vaucouleurs on the raw data  
 pre-exponential factor = -26108.8827771398 (+/-) 159.5723836325  
 rate constant = 3.7826 (+/-) 195288.7901  
 yintercept = 27750.4116 (+/-) 585414854.0087  
 de Vaucouleurs on the raw data  
 pre-exponential factor = -11869.9461095652 (+/-) 67.6496230803  
 rate constant = 4.1330 (+/-) 140862.2781  
 yintercept = 14356.0015 (+/-) 175697720.5452



## Curve Fitting - Residual Analysis - Radial Profile Curve Fitting - Exponential Curve

In [6]:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy as scipy
4 from astropy.stats import sigma_clipped_stats
5 from scipy import stats
6 import warnings
7 warnings.filterwarnings('ignore')
8
9 def exponential(x, a, k, b):
10 return a*np.log10(x*k) + b
11
12 fig = plt.figure()
13
14 data = np.genfromtxt("M87RedPlotValues.tsv", delimiter="\t", skip_header=1)
15
16 x = (data[:,0])
17 y = (data[:,1])
18
19 mean, median, std = sigma_clipped_stats(y)
20 #print(min(y),max(y),mean,median,std)
21 #print(x,y)
22 #print(min(x),max(x))
23 #print(x.shape)
24
25 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(exponential, x, y)
26 perr_exponential = np.sqrt(np.diag(pcov_exponential))
27 print ("Exponential on the raw data")
28 #print (popt_exponential, pcov_exponential)
29 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponential[0], perr_exponential[0]))
30 print ("rate constant = %0.4f (+/-) %0.4f" % (popt_exponential[1], perr_exponential[1]))
31 print ("yintercept = %0.4f (+/-) %0.4f" % (popt_exponential[2], perr_exponential[2]))
32
33 xtest = np.linspace(min(x),max(x),3969)
34 testy = (popt_exponential[0]*np.log10(xtest*popt_exponential[1])) + popt_exponential[2]
35 diffy = (popt_exponential[0]*np.log10(x*popt_exponential[1])) + popt_exponential[2]
36
37 #plt.subplot(2, 2, 1)
38 frame1=fig.add_axes((.1,.3,.8,.6))
39 plt.plot(x, y, linestyle = 'None', marker = 'o')
40 plt.plot(xtest,testy)
41 #plt.ylim(12000,18000)
42 plt.ylabel('Counts')
43 plt.title('M87 Red Exponential')
44
45
46 #Residual plot
47 difference = y - diffy
48 frame2=fig.add_axes((.1,.1,.8,.2))
49 plt.plot(x,difference,'or')
```

50

```
51 plt.xlabel('Pixels')
52 plt.show()
53
54 fig = plt.figure()
55
56 data = np.genfromtxt("M87BluePlotValues.tsv", delimiter="\t", skip_header=1)
57
58 #x, y = xy.T
59 x = (data[:,0])
60 y = (data[:,1])
61
62 #x, y = xy.T
63 mean, median, std = sigma_clipped_stats(y)
64 #print(min(y),max(y),mean,median,std)
65 #print(min(x),max(x))
66 #print(x,y)
67 #print(x.shape)
68
69 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(exponential, x, y)
70 perr_exponential = np.sqrt(np.diag(pcov_exponential))
71 print ("Exponential on the raw data")
72 #print (popt_exponential, pcov_exponential)
73 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponential[0], perr_exponential[0]))
74 print ("rate constant = %0.4f (+/-) %0.4f" % (popt_exponential[1], perr_exponential[1]))
75 print ("yintercept = %0.4f (+/-) %0.4f" % (popt_exponential[2], perr_exponential[2]))
76
77 xtest = np.linspace(min(x),max(x),3969)
78 testy = (popt_exponential[0]*np.log10(xtest*popt_exponential[1])) + popt_exponential[2]
79 diffy = (popt_exponential[0]*np.log10(x*popt_exponential[1])) + popt_exponential[2]
80
81 #plt.subplot(2, 2, 2)
82 frame1=fig.add_axes((.1,.3,.8,.6))
83 plt.plot(x, y, linestyle = 'None', marker = 'o')
84 plt.plot(xtest,testy)
85 #plt.ylim(12000,18000)
86 plt.ylabel('Counts')
87 plt.title('M87 Blue Exponential')
88
89
90 #Residual plot
91 difference = y - diffy
92 frame2=fig.add_axes((.1,.1,.8,.2))
93 plt.plot(x,difference,'or')
94
95 plt.xlabel('Pixels')
96 plt.show()
97
98 fig = plt.figure()
99
100 data = np.genfromtxt("M81RedPlotValues.tsv", delimiter="\t", skip_header=1)
```

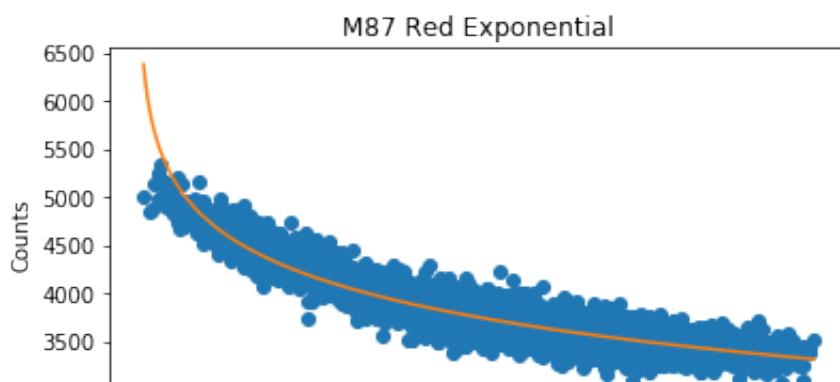
```
101
102 #x, y = xy.T
103 x = (data[:,0])
104 y = (data[:,1])
105
106 #x, y = xy.T
107 #mean, median, std = sigma_clipped_stats(y)
108 #print(min(y),max(y),mean,median,std)
109 #print(min(x),max(x))
110 #print(x,y)
111 #print(x.shape)
112
113 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(exponen
114 perr_exponential = np.sqrt(np.diag(pcov_exponential))
115 print ("Exponential on the rax data")
116 #print (popt_exponential, pcov_exponential)
117 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponen
118 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
119 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
120
121 xtest = np.linspace(min(x),max(x),3969)
122 testy = (popt_exponential[0]*np.log10(xtest*popt_exponential[1])) + p
123 diffy = (popt_exponential[0]*np.log10(x*popt_exponential[1])) + popt_
124
125 #plt.subplot(2, 2, 3)
126 frame1=fig.add_axes((.1,.3,.8,.6))
127 plt.plot(x, y, linestyle = 'None', marker = 'o')
128 plt.plot(xtest,testy)
129 #plt.ylim(15000,25000)
130 plt.ylabel('Counts')
131 plt.title('M81 Red Exponential')
132
133
134 #Residual plot
135 difference = y - diffy
136 frame2=fig.add_axes((.1,.1,.8,.2))
137 plt.plot(x,difference,'or')
138
139 plt.xlabel('Pixels')
140 plt.show()
141
142
143 fig = plt.figure()
144
145 data = np.genfromtxt("M81BluePlotValues.tsv", delimiter="\t", skip_l
146
147 #x, y = xy.T
148 x = (data[:,0])
149 y = (data[:,1])
150
151 mean, median, std = sigma_clipped_stats(v)
```

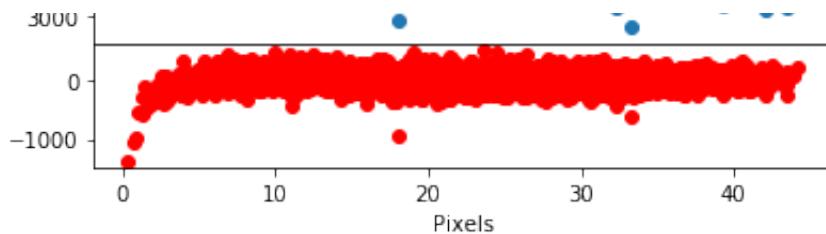
```

--> mean, median, std = np.percentile(x),
152 #print(min(y),max(y),mean,median,std)
153 #print(min(x),max(x))
154 #print(x,y)
155 #print(x.shape)
156
157 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(exponen
158 perr_exponential = np.sqrt(np.diag(pcov_exponential))
159 print ("Exponential on the raw data")
160 #print (popt_exponential, pcov_exponential)
161 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponentia
162 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], perr_expon
163 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_exponentia
164
165 xtest = np.linspace(min(x),max(x),3969)
166 testy = (popt_exponential[0]*np.log10(xtest*popt_exponential[1])) + popt_exponentia
167 diffy = (popt_exponential[0]*np.log10(x*popt_exponential[1])) + popt_exponentia
168
169 #plt.subplot(2, 2, 4)
170 frame1=fig.add_axes((.1,.3,.8,.6))
171 plt.plot(x, y, linestyle = 'None', marker = 'o')
172 plt.plot(xtest,testy)
173 #plt.ylim(15000,25000)
174 plt.ylabel('Counts')
175 plt.title('M81 Blue Exponential')
176
177
178 #Residual plot
179 difference = y - diffy
180 frame2=fig.add_axes((.1,.1,.8,.2))
181 plt.plot(x,difference,'or')
182
183 plt.xlabel('Pixels')
184 #plt.set_yticklabels([])
185 plt.show()

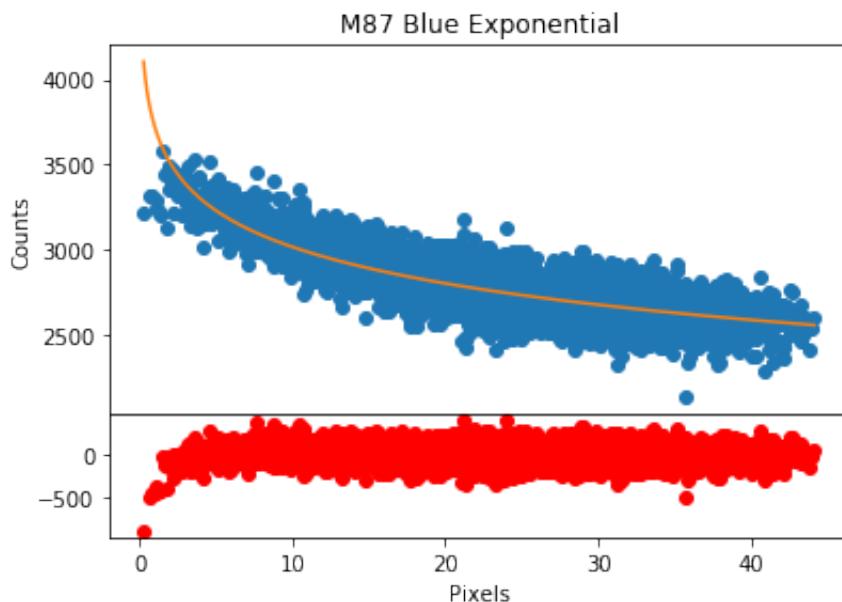
```

Exponential on the raw data  
pre-exponential factor = -1459.7199939749 (+/-) 9.6901457507  
rate constant = 10610.8435 (+/-) 496067472.4748  
yintercept = 11591.0500 (+/-) 29611851.6830

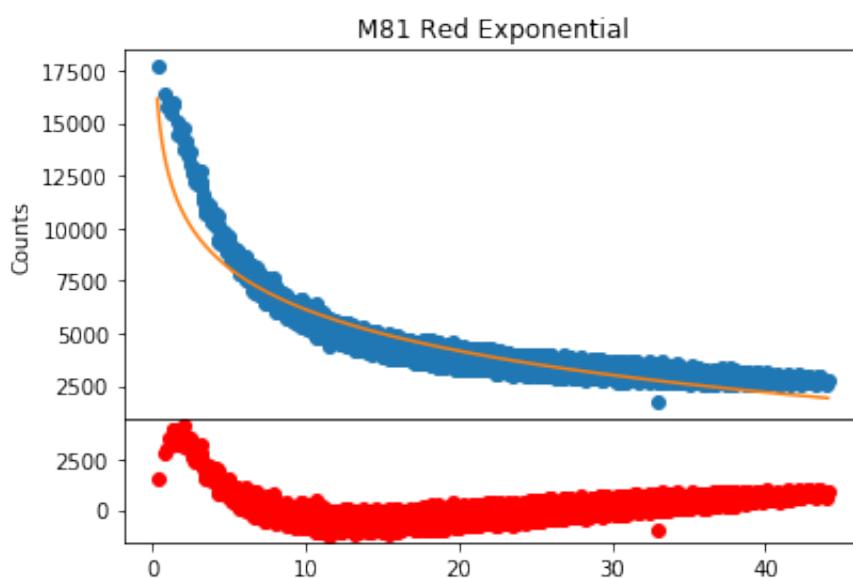




Exponential on the raw data  
 pre-exponential factor = -714.8465042582 (+/-) 7.4201598756  
 rate constant = 986.1767 (+/-) 68607035.7887  
 yintercept = 5870.2950 (+/-) 21597873.6918

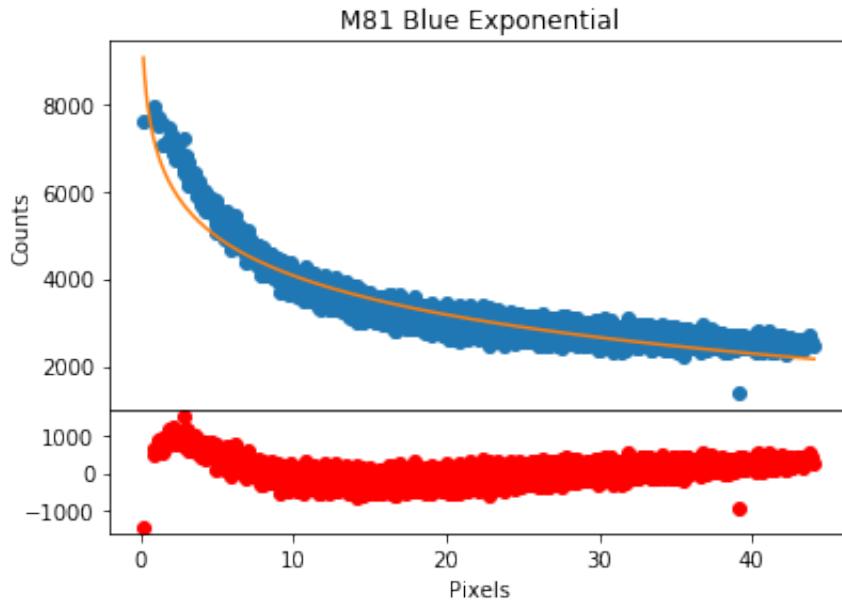


Exponential on the rax data  
 pre-exponential factor = -6527.2206979377 (+/-) 39.8917972998  
 rate constant = 2.3298 (+/-) 393289.5814  
 yintercept = 15062.6515 (+/-) 477989226.6355



Pixels

```
Exponential on the rax data
pre-exponential factor = -2967.4865363590 (+/-) 16.9161690243
rate constant = 14.0544 (+/-) 1025358.7545
yintercept = 10447.1207 (+/-) 93914381.9431
```



## Curve Fitting - Residual Analysis - Radial Profile Curve Fitting - de Vaucouleurs Curve

In [12]:

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy as scipy
4 from astropy.stats import sigma_clipped_stats
5 from scipy import stats
6
7 #def exponential(x, a, k, b):
8 # return a*np.log10(x*k) + b
9 def SecondOrder(x, a, k, b):
10 return a*np.log10(k*x**.25) + b
11
12
13 fig = plt.figure()
14
15 data = np.genfromtxt("M87RedPlotValues.tsv", delimiter="\t", skip_he
16
17 x = (data[:,0])
18 y = (data[:,1])
19
20 mean, median, std = sigma_clipped_stats(y)
21
```

```
21 #print(min(y),max(y),mean,median,std)
22 #print(x,y)
23 #print(min(x),max(x))
24 #print(x.shape)
25
26 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
27 perr_exponential = np.sqrt(np.diag(pcov_exponential))
28 print ("de Vaucouleurs on the raw data")
29 #print (popt_exponential, pcov_exponential)
30 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponentia
31 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
32 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
33
34 xtest = np.linspace(min(x),max(x))
35 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**2
36 diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x**.25)
37
38 #plt.subplot(2, 2, 1)
39 frame1=fig.add_axes((.1,.3,.8,.6))
40 plt.plot(x, y, linestyle = 'None', marker = '.')
41 plt.plot(xtest,testy)
42 #plt.ylim(12000,18000)
43 plt.ylabel('Counts')
44 plt.title('M87 Red de Vaucouleurs')
45
46 #Residual plot
47 difference = y - diffy
48 frame2=fig.add_axes((.1,.1,.8,.2))
49 plt.plot(x,difference,'or', marker = '.')
50
51 plt.xlabel('Pixels')
52 plt.show()
53
54 fig = plt.figure()
55
56 data = np.genfromtxt("M87BluePlotValues.tsv", delimiter="\t", skip_l
57
58 #x, y = xy.T
59 x = (data[:,0])
60 y = (data[:,1])
61
62 #x, y = xy.T
63 mean, median, std = sigma_clipped_stats(y)
64 #print(min(y),max(y),mean,median,std)
65 #print(min(x),max(x))
66 #print(x,y)
67 #print(x.shape)
68
69 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
70 perr_exponential = np.sqrt(np.diag(pcov_exponential))
71 print ("de Vaucouleurs on the raw data")
```

```
72 #print (popt_exponential, pcov_exponential)
73 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponentia
74 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], perr_
75 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
76
77 xtest = np.linspace(min(x),max(x))
78 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.
79 diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x**.25))
80
81 #plt.subplot(2, 2, 2)
82 frame1=fig.add_axes((.1,.3,.8,.6))
83 plt.plot(x, y, linestyle = 'None', marker = '.')
84 plt.plot(xtest,testy)
85 #plt.ylim(12000,18000)
86 plt.ylabel('Counts')
87 plt.title('M87 Blue de Vaucouleurs')
88
89 #Residual plot
90 difference = y - diffy
91 frame2=fig.add_axes((.1,.1,.8,.2))
92 plt.plot(x,difference,'or', marker = '.')
93
94 plt.xlabel('Pixels')
95 plt.show()
96
97 fig = plt.figure()
98
99 data = np.genfromtxt("M81RedPlotValues.tsv", delimiter="\t", skip_he
100
101 #x, y = xy.T
102 x = (data[:,0])
103 y = (data[:,1])
104
105 #x, y = xy.T
106 mean, median, std = sigma_clipped_stats(y)
107 #print(min(y),max(y),mean,median,std)
108 #print(min(x),max(x))
109 #print(x,y)
110 #print(x.shape)
111
112 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
113 perr_exponential = np.sqrt(np.diag(pcov_exponential))
114 print ("de Vaucouleurs on the rax data")
115 #print (popt_exponential, pcov_exponential)
116 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponentia
117 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], perr_
118 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
119
120 xtest = np.linspace(min(x),max(x))
121 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.
```

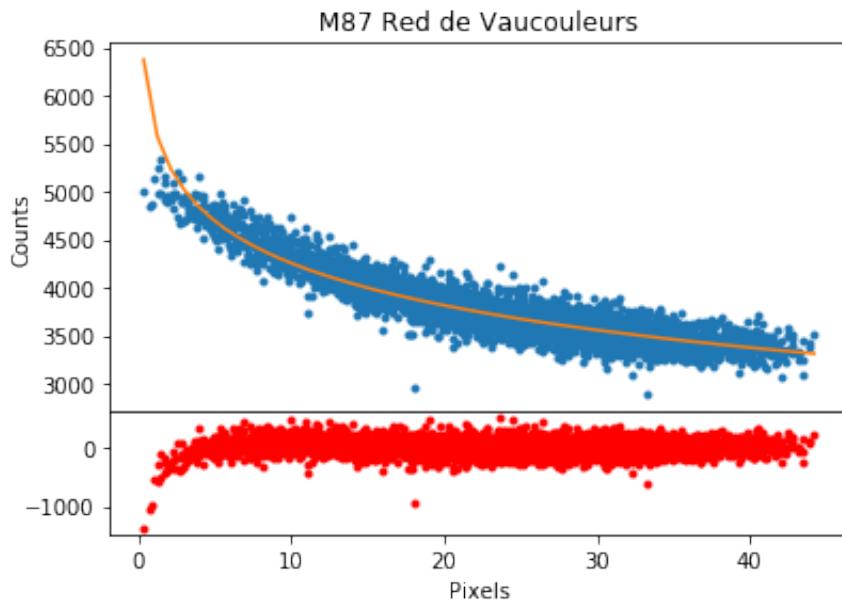
```
122 diffy = (popt_exponential[0]*np.log10((popt_exponential[1])**x**.25))
123
124 #plt.subplot(2, 2, 3)
125 frame1=fig.add_axes((.1,.3,.8,.6))
126 plt.plot(x, y, linestyle = 'None', marker = '.')
127 plt.plot(xtest,testy)
128 #plt.ylim(15000,25000)
129 plt.ylabel('Counts')
130 plt.title('M81 Red de Vaucouleurs')
131
132 #Residual plot
133 difference = y - diffy
134 frame2=fig.add_axes((.1,.1,.8,.2))
135 plt.plot(x,difference,'or', marker = '.')
136
137 plt.xlabel('Pixels')
138 plt.show()
139
140
141 fig = plt.figure()
142
143 data = np.genfromtxt("M81BluePlotValues.tsv", delimiter="\t", skip_l
144
145 #x, y = xy.T
146 x = (data[:,0])
147 y = (data[:,1])
148
149 mean, median, std = sigma_clipped_stats(y)
150 #print(min(y),max(y),mean,median,std)
151 #print(min(x),max(x))
152 #print(x,y)
153 #print(x.shape)
154
155 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
156 perr_exponential = np.sqrt(np.diag(pcov_exponential))
157 print ("de Vaucouleurs on the rax data")
158 #print (popt_exponential, pcov_exponential)
159 print ("pre-exponential factor = %0.10f (+/-) %0.10f" %(popt_exponen
160 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
161 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
162
163 xtest = np.linspace(min(x),max(x))
164 testy = (popt_exponential[0]*np.log10((popt_exponential[1])**xtest**.
165 diffy = (popt_exponential[0]*np.log10((popt_exponential[1])**x**.25))
166
167 #plt.subplot(2, 2, 4)
168 frame1=fig.add_axes((.1,.3,.8,.6))
169 plt.plot(x, y, linestyle = 'None', marker = '.')
170 plt.plot(xtest,testy)
171 #plt.ylim(15000,25000)
172 plt.ylabel('Counts')
```

```

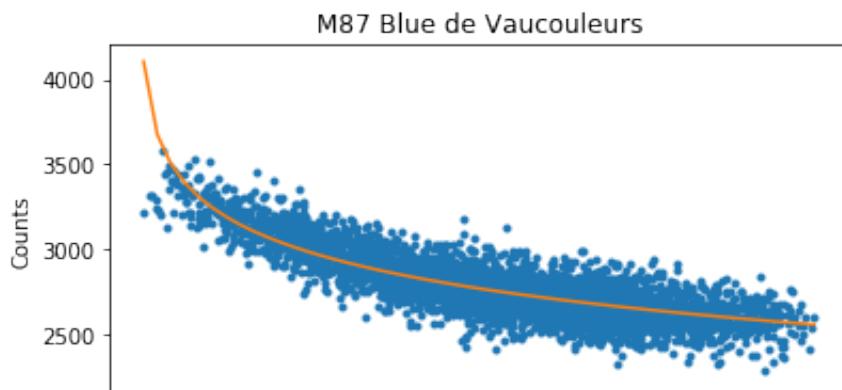
172 plt.ylabel('Counts')
173 plt.title('M81 Blue de Vaucouleurs')
174
175
176 #Residual plot
177 difference = y - diffy
178 frame2=fig.add_axes((.1,.1,.8,.2))
179 plt.plot(x,difference,'or', marker = '.')
180
181 plt.xlabel('Pixels')
182 #plt.set_yticklabels([])
183 plt.show()

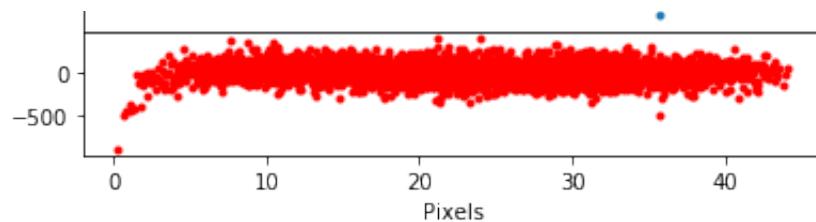
```

de Vaucouleurs on the raw data  
 pre-exponential factor = -5838.8799644059 (+/-) 38.7672592974  
 rate constant = 30.2440 (+/-) 1383994.1081  
 yintercept = 14359.8604 (+/-) 116040309.3519

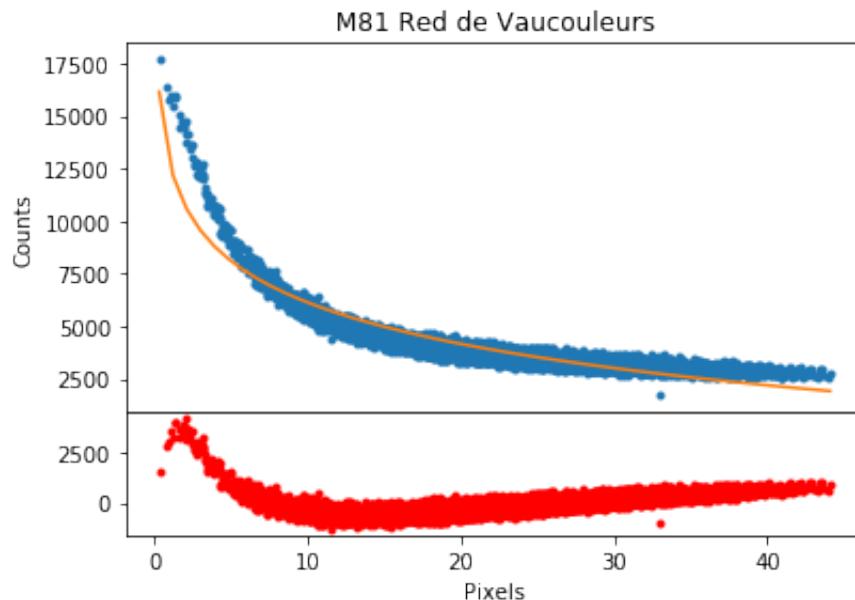


de Vaucouleurs on the raw data  
 pre-exponential factor = -2859.3860296192 (+/-) 29.6808886044  
 rate constant = 45.3364 (+/-) 1777622.6717  
 yintercept = 8466.4980 (+/-) 48691125.9814

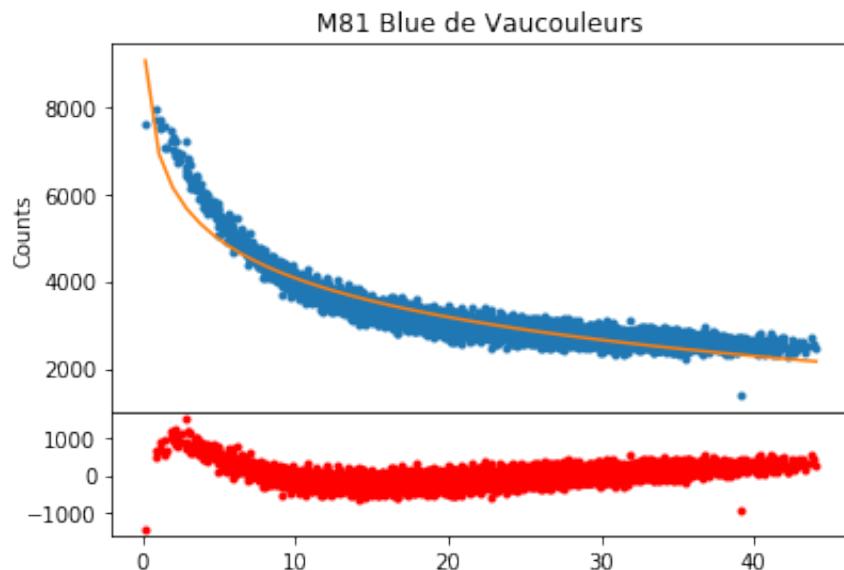




de Vaucouleurs on the rax data  
pre-exponential factor = -26108.8827771398 (+/-) 159.5723836325  
rate constant = 3.7826 (+/-) 195288.7901  
yintercept = 27750.4116 (+/-) 585414854.0087



de Vaucouleurs on the rax data  
pre-exponential factor = -11869.9461095652 (+/-) 67.6496230803  
rate constant = 4.1330 (+/-) 140862.2781  
yintercept = 14356.0015 (+/-) 175697720.5452



Pixels

# Curve Fitting - Residual Analysis - Radial Profile Curve Fitting - R-B Filters

In [15]:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy as scipy
4 from astropy.stats import sigma_clipped_stats
5 from scipy import stats
6
7 def exponential(x, a, k, b):
8 return a*np.log10(x*k) + b
9 def SecondOrder(x, a, k, b):
10 return a*np.log10(k*x**.25) + b
11
12
13 fig = plt.figure()
14
15 data = np.genfromtxt("M87R_BStackWCSPlotValues.tsv", delimiter="\t")
16
17 x = (data[:,0])
18 y = (data[:,1])
19
20 mean, median, std = sigma_clipped_stats(y)
21 #print(min(y),max(y),mean,median,std)
22 #print(x,y)
23 #print(min(x),max(x))
24 #print(x.shape)
25
26 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(SecondOrder, x, y)
27 perr_exponential = np.sqrt(np.diag(pcov_exponential))
28 print ("de Vaucouleurs on the raw data")
29 #print (popt_exponential, pcov_exponential)
30 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponential[0], perr_exponential[0]))
31 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], perr_exponential[1]))
32 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_exponential[2]))
33
34 xtest = np.linspace(min(x),max(x))
35 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.25))
36 diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x**.25))
37
38 #plt.subplot(2, 2, 1)
39 frame1=fig.add_axes((.1,.3,.8,.6))
40 plt.plot(x, y, linestyle = 'None', marker = '.')
41 plt.plot(xtest,testy)
42 #plt.ylim(12000,18000)
43 plt.ylabel('Counts')
```

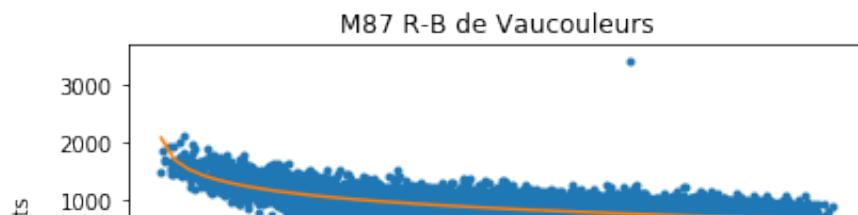
```
44 plt.title('M87 R-B de Vaucouleurs')
45
46
47 #Residual plot
48 difference = y - diffy
49 frame2=fig.add_axes((.1,.1,.8,.2))
50 plt.plot(x,difference,'or', marker = '.')
51
52 plt.xlabel('Pixels')
53 plt.show()
54
55 fig = plt.figure()
56
57 data = np.genfromtxt("M87R_BStackWCSPPlotValues.tsv", delimiter="\t")
58
59 #x, y = xy.T
60 x = (data[:,0])
61 y = (data[:,1])
62
63 #x, y = xy.T
64 mean, median, std = sigma_clipped_stats(y)
65 #print(min(y),max(y),mean,median,std)
66 #print(min(x),max(x))
67 #print(x,y)
68 #print(x.shape)
69
70 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(exponen
71 perr_exponential = np.sqrt(np.diag(pcov_exponential))
72 print ("Exponential on the raw data")
73 #print (popt_exponential, pcov_exponential)
74 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponent
75 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
76 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
77
78 xtest = np.linspace(min(x),max(x))
79 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +
80 diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x) + popt_
81
82 #plt.subplot(2, 2, 2)
83 frame1=fig.add_axes((.1,.3,.8,.6))
84 plt.plot(x, y, linestyle = 'None', marker = '.')
85 plt.plot(xtest,testy)
86 #plt.ylim(12000,18000)
87 plt.ylabel('Counts')
88 plt.title('M87 R-B Exponential')
89
90
91 #Residual plot
92 difference = y - diffy
93 frame2=fig.add_axes((.1,.1,.8,.2))
94 plt.plot(x,difference,'or', marker = '.')
```

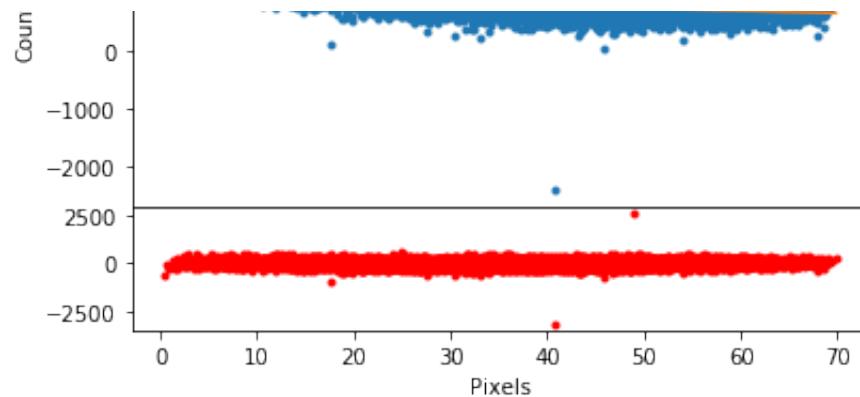
```
 plt.plot(x,difference, 'or', marker = '.',
95
96 plt.xlabel('Pixels')
97 plt.show()
98
99 fig = plt.figure()
100
101 data = np.genfromtxt("M81R_BStackWCSPlotValues.tsv", delimiter="\t")
102
103 #x, y = xy.T
104 x = (data[:,0])
105 y = (data[:,1])
106
107 #x, y = xy.T
108 mean, median, std = sigma_clipped_stats(y)
109 #print(min(y),max(y),mean,median,std)
110 #print(min(x),max(x))
111 #print(x,y)
112 #print(x.shape)
113
114 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(Second
115 perr_exponential = np.sqrt(np.diag(pcov_exponential))
116 print ("de Vaucouleurs on the rax data")
117 #print (popt_exponential, pcov_exponential)
118 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_exponent
119 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
120 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
121
122 xtest = np.linspace(min(x),max(x))
123 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**
124 diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x**.25))
125
126 #plt.subplot(2, 2, 3)
127 frame1=fig.add_axes((.1,.3,.8,.6))
128 plt.plot(x, y, linestyle = 'None', marker = '.')
129 plt.plot(xtest,testy)
130 #plt.ylim(15000,25000)
131 plt.ylabel('Counts')
132 plt.title('M81 R-B de Vaucouleurs')
133
134
135 #Residual plot
136 difference = y - diffy
137 frame2=fig.add_axes((.1,.1,.8,.2))
138 plt.plot(x,difference,'or', marker = '.')
139
140 plt.xlabel('Pixels')
141 plt.show()
142
143
144 fig = plt.figure()
```

```

145
146 data = np.genfromtxt("M81R_BStackWCSPlotValues.tsv", delimiter="\t")
147 #data = np.genfromtxt("M81RedPlotValues.tsv", delimiter="\t", skip_1
148
149 #x, y = xy.T
150 x = (data[:,0])
151 y = (data[:,1])
152
153 mean, median, std = sigma_clipped_stats(y)
154 #print(min(y),max(y),mean,median,std)
155 #print(min(x),max(x))
156 #print(x,y)
157 #print(x.shape)
158
159 popt_exponential, pcov_exponential = scipy.optimize.curve_fit(expon
160 perr_exponential = np.sqrt(np.diag(pcov_exponential))
161 print ("Exponential on the rax data")
162 #print (popt_exponential, pcov_exponential)
163 print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expon
164 print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
165 print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_
166
167 xtest = np.linspace(min(x),max(x))
168 testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +
169 diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x) + pop
170
171 #plt.subplot(2, 2, 4)
172 frame1=fig.add_axes((.1,.3,.8,.6))
173 plt.plot(x, y, linestyle = 'None', marker = '.')
174 plt.plot(xtest,testy)
175 #plt.ylim(15000,25000)
176 plt.ylabel('Counts')
177 plt.title('M81 R-B Exponential')
178
179
180 #Residual plot
181 difference = y - diffy
182 frame2=fig.add_axes((.1,.1,.8,.2))
183 plt.plot(x,difference,'or', marker = '.')
184
185 plt.xlabel('Pixels')
186 #plt.set_yticklabels([])
187 plt.show()

```





Exponential on the raw data

## References

- Collins, K. (2017). AstrolImageJ: Image Processing and Photometric Extraction for Ultra-Precise Astronomical Light Curves (Expanded Edition) arXiv:1701.04817v1
- de Vaucouleurs, G. (1948). Ann. d'Astrophys. 11, 247.
- Wilmer, C. (2018). The Absolute Magnitude of the Sun in Several Filters, The Astrophysical Journal Supplement Series, Volume 236, Number 2