Lab 4 Radial Profiles of Galaxies

Patrick Selep

Abstract

Several galaxies were observed in both red and blue filters. Examples of both spiral and eliptical 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 ot 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 Phython 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 galazy.

Spiral and Eliptical Galaxies

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

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 eliptical.

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 eliptical galaxy the concentraction 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 aperature 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 attibuted 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 eliptical 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

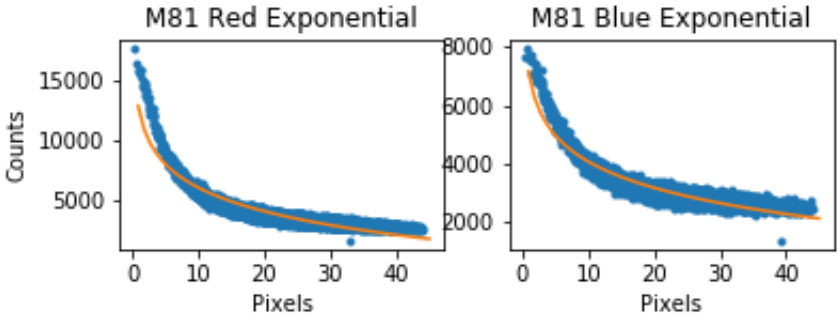
$$I(R) = I_0 exp(-kR^{1/4})$$

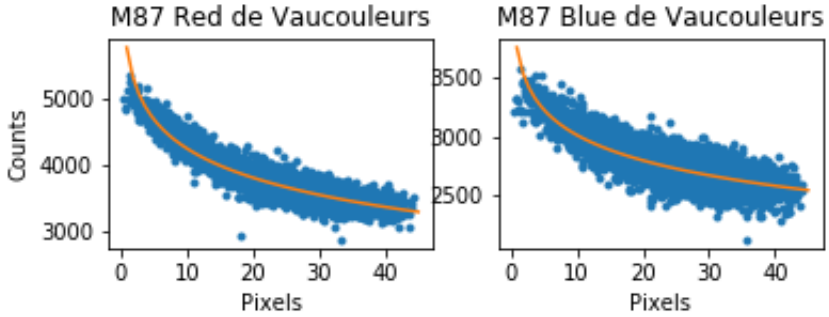
Absolute Magnitude

Absolute Magnitude, M, was calculated as follow:

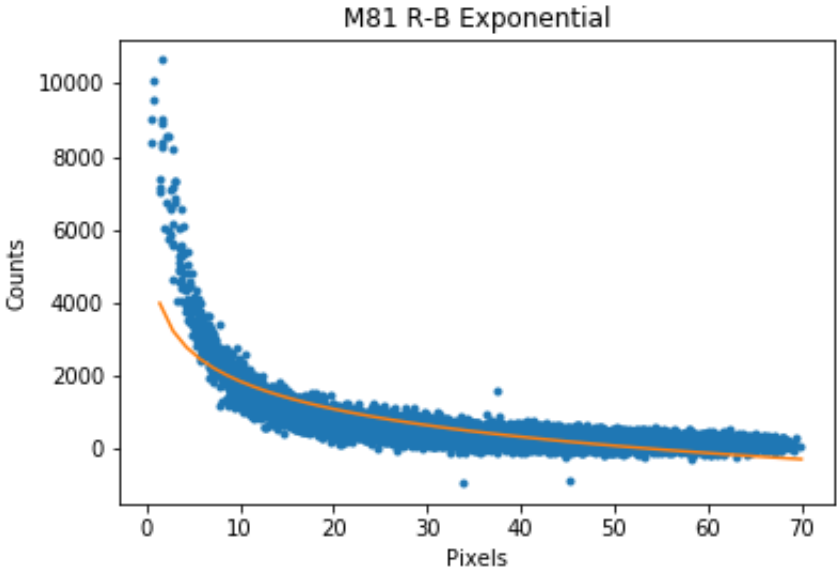
$$M = m - 2.5log_{10}(d/10)^2$$

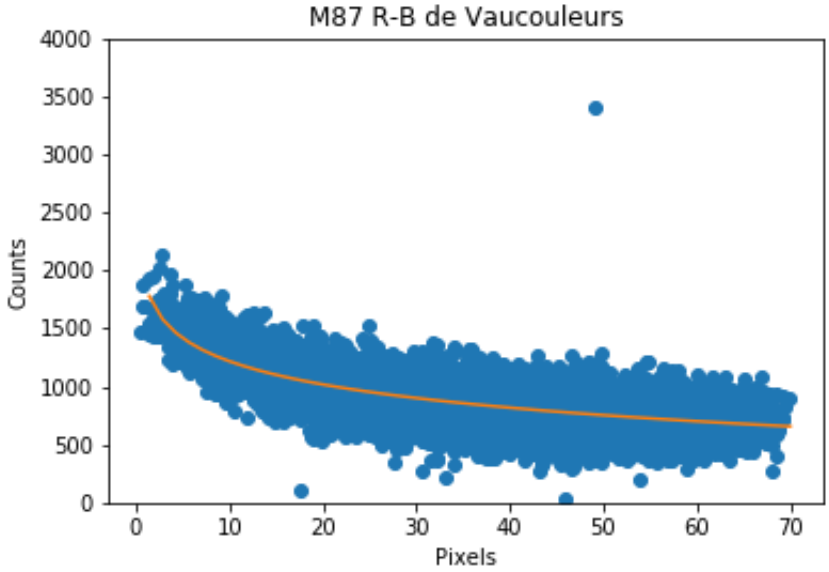
Individual filters


M81 Red Exponential
M81 Blue Exponential
Counts
Pixels
Pixels


M87 Red de Vaucouleurs
M87 Blue de Vaucouleurs
Counts
Pixels
Pixels

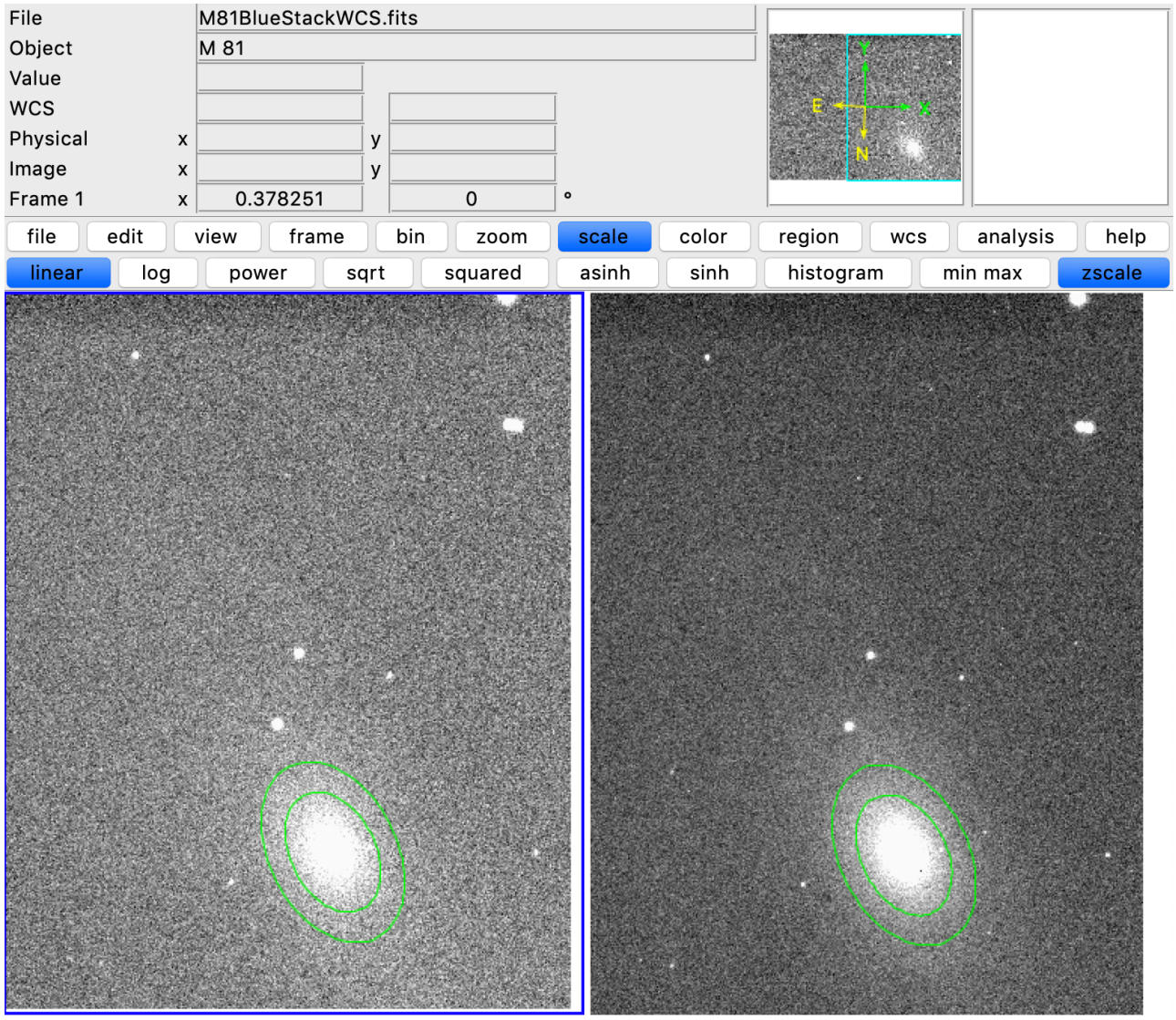
R-B Filters





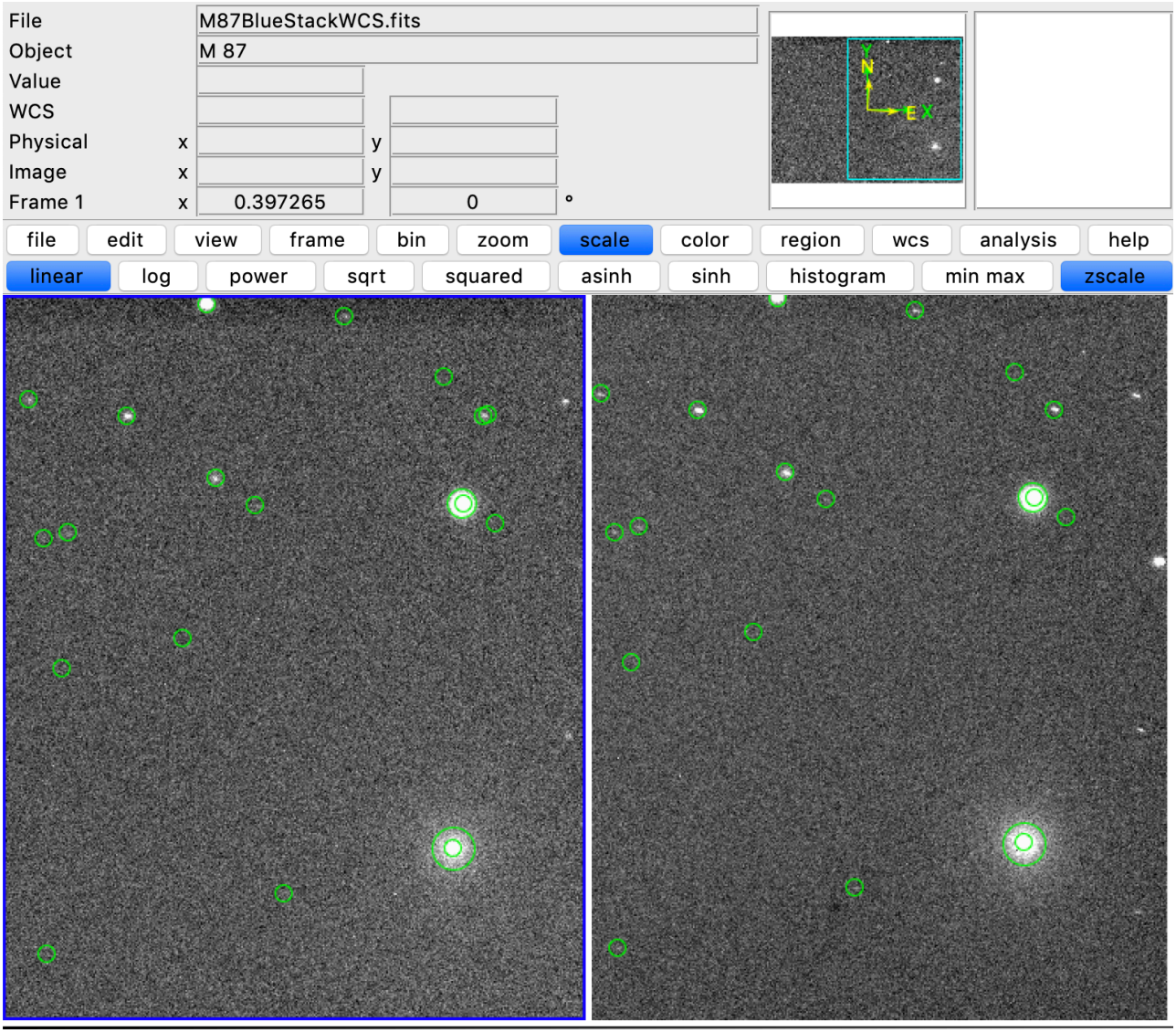
Total Luminosity

M81



Bmag	Vmag	rmag	i
11.938	11.349	11.237	

M87



Bmag	Vmag	rmag
9.693	9.297	9.005

In [28]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as scypy
from astropy.stats import sigma_clipped_stats
from scipy import stats

Msunb = 5.44    # Wilmer, C. (2018)
mRefb = 11.938
```

```
f81b = 69406464
fRefb = 2026251
m81b = mRefb - 2.5 * np.log10(f81b/fRefb)
DistM81 = 3.6 * 10**6
M81b = 5 + m81b - 5 * np.log10(DistM81)
Suns81b = 100**((Msunb-M81b)/5)

print("Apparent magnitude of M81 in Blue: %0.3f"% (m81b))
print("Absolute magnitude of M81 in Blue: %0.3f"% (M81b))
print("In terms of suns: %0.2f billion"%(Suns81b/10**9))
print()

Msunr = 4.43  # Wilmer, C. (2018)
mRefr = 11.237
f81r = 76651327
fRefr = 1526263
m81r = mRefr - 2.5 * np.log10(f81r/fRefr)
M81r = 5 + m81r - 5 * np.log10(DistM81)
Suns81r = 100**((Msunr-M81r)/5)

print("Apparent magnitude of M81 in Red: %0.3f"% (m81r))
print("Absolute magnitude of M81 in Red: %0.3f"% (M81r))
print("In terms of suns: %0.2f billion"%(Suns81r/10**9))
print()

mRefb = 9.693
fRefb = 12774613
f87b = 11496322
m87b = mRefb - 2.5 * np.log10(f87b/fRefb)
DistM87 = 16.4 * 10**6
M87b = 5 + m87b - 5 * np.log10(DistM87)
Suns87b = 100**((Msunb-M87b)/5)

print("Apparent magnitude of M87 in Blue: %0.3f"% (m87b))
print("Absolute magnitude of M87 in Blue: %0.3f"% (M87b))
print("In terms of suns: %0.2f billion"%(Suns87b/10**9))
print()

mRefr = 9.005
f87r = 15572807
fRefr = 16062810
m87r = mRefr - 2.5 * np.log10(f87r/fRefr)
M87r = 5 + m87r - 5 * np.log10(DistM87)
Suns87r = 100**((Msunr-M87r)/5)

print("Apparent magnitude of M87 in Red: %0.3f"% (m87r))
print("Absolute magnitude of M87 in Red: %0.3f"% (M87r))
print("In terms of suns: %0.2f billion"%(Suns87r/10**9))
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

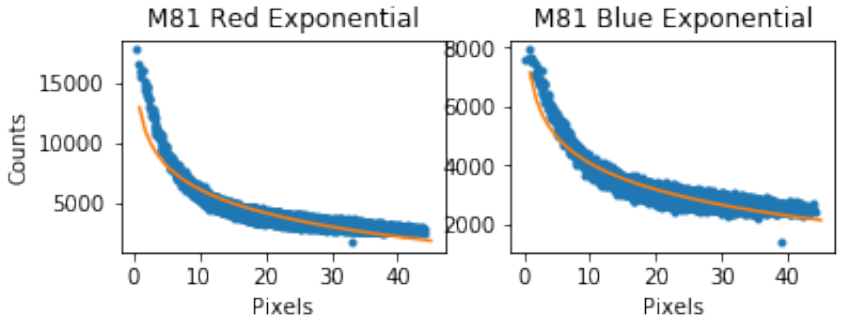
In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as scypy
from astropy.stats import sigma_clipped_stats
from scipy import stats
import warnings
warnings.filterwarnings('ignore')


def exponential(x, a, k, b):
    return a*np.log10(x*k) + b

#def SecondOrder(x, a, k, b):
#    return a*np.log10(k*x**.25) + b

fig = plt.figure()

data = np.genfromtxt("M81RedPlotValues.tsv", delimiter="\t", skip_he

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])


mean, median, std = sigma_clipped_stats(y)
```

```
#print(min(y),max(y),mean,median,std)
#print(x,y)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(expone
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("Exponential on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(0,45)
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +

plt.subplot(2, 2, 1)
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(10000,70000)
plt.title('M81 Red Exponential')
plt.xlabel('Pixels')
plt.ylabel('Counts')
#plt.show()

data = np.genfromtxt("M81BluePlotValues.tsv", delimiter="\t", skip_h

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])

#x, y = xy.T
mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(x,y)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(expone
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("Exponential on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(0,45)
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +

plt.subplot(2, 2, 2)
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(10000,70000)
plt.title('M81 Blue Exponential')
plt.xlabel('Pixels')
```

```
plt.xlabel('Pixels')
#plt.ylabel('Counts')
plt.savefig("M81exponential.png")
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

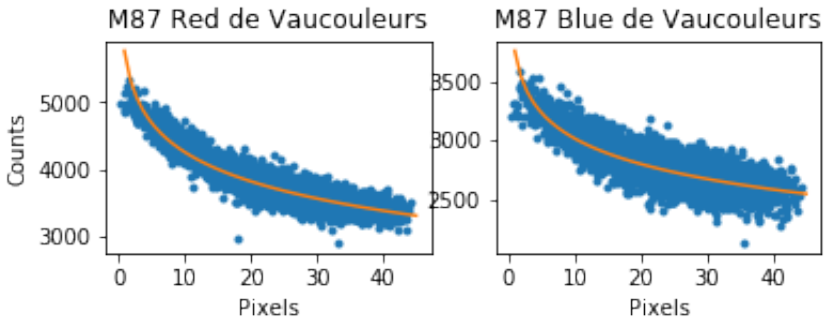
In [4]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as scypy
from astropy.stats import sigma_clipped_stats
from scipy import stats
import warnings
warnings.filterwarnings('ignore')


#def exponential(x, a, k, b):
#    return a*np.log10(x*k) + b

def SecondOrder(x, a, k, b):
    return a*np.log10(k*x**.25) + b

fig = plt.figure()

data = np.genfromtxt("M87RedPlotValues.tsv", delimiter="\t", skip_he

#x, y = xy.T
x = (data[:,0])
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 = scypy.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 = scypy.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 = '.')
```

```
plt.plot(xtest,testy)
#plt.ylim(10000,70000)
plt.title('M87 Blue de Vaucouleurs')
plt.xlabel('Pixels')
#plt.ylabel('Counts')
plt.savefig("M87deVaucouleurs.png")
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

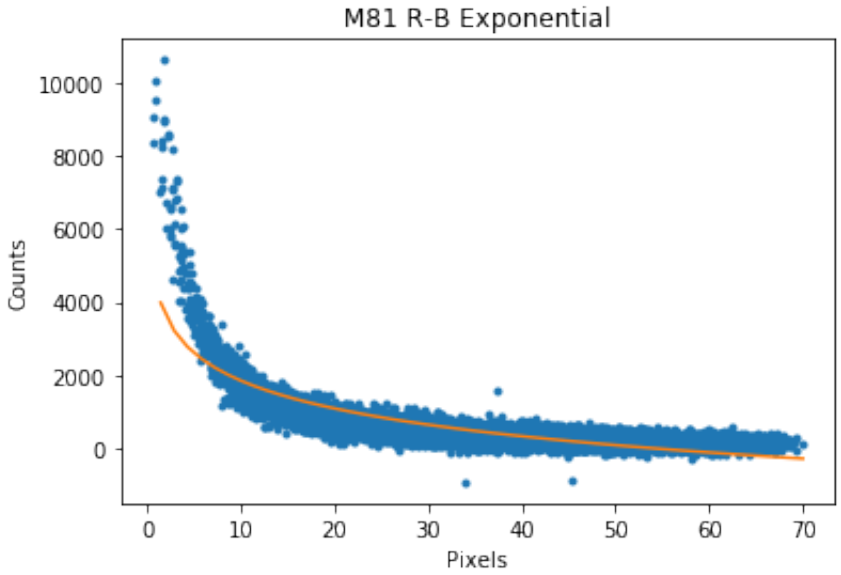
In [8]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as scypy
from astropy.stats import sigma_clipped_stats
from scipy import stats

def exponential(x, a, k, b):
    return a*np.log10(x*k) + b

#def SecondOrder(x, a, k, b):
#    return a*np.log10(k*x**.25) + b

fig = plt.figure()

data = np.genfromtxt("M81R_BStackWCSPlotValues.tsv", delimiter="\t",

#x, y = xy.T
x = (data[:,0])
```

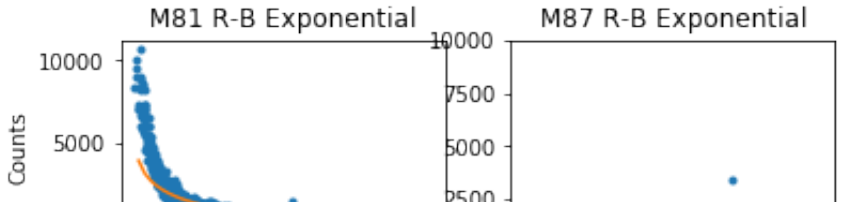
```
y = (data[:,1])


mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(x,y)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(expone
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("Exponential on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(0,70)
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +

plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(0,4000)
plt.title('M81 R-B Exponential')
plt.xlabel('Pixels')
plt.ylabel('Counts')
plt.savefig("M81R_BExponential.png")
plt.show()

plt.subplot(2, 2, 1)
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(10000,70000)
plt.title('M81 R-B Exponential')
plt.xlabel('Pixels')
plt.ylabel('Counts')
#plt.show()

data = np.genfromtxt("M87R_BStackWCSPlotValues.tsv", delimiter="\t",

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])

#x, y = xy.T
mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(x,y)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(expone
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("Exponential on the raw data")
#print (popt_exponential, pcov_exponential)
```

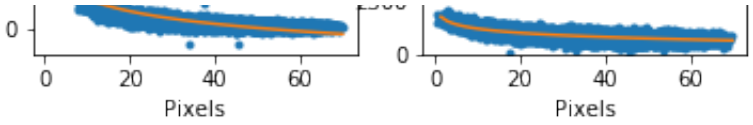
```
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(0,70)
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +

plt.subplot(2, 2, 2)
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
plt.ylim(0,10000)
plt.title('M87 R-B Exponential')
plt.xlabel('Pixels')
#plt.ylabel('Counts')
plt.show()


```

```
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

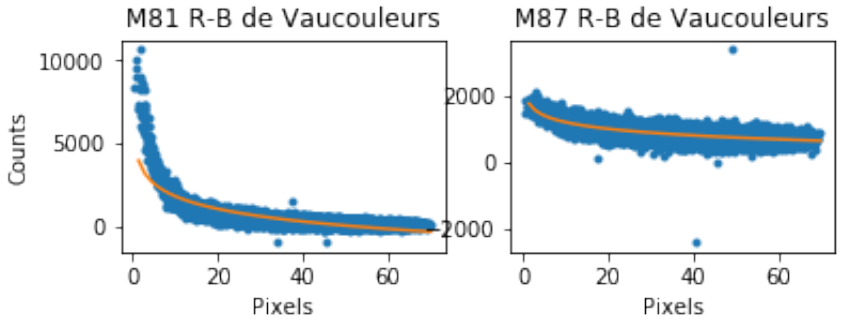
In [9]:

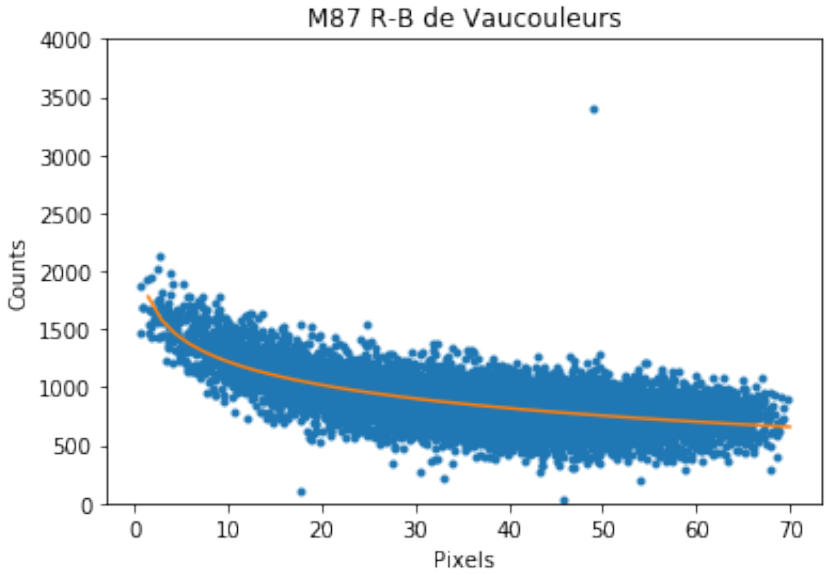
```
import numpy as np
import matplotlib.pyplot as plt
import scipy as scypy
from astropy.stats import sigma_clipped_stats
from scipy import stats

#def exponential(x, a, k, b):
#    return a*np.log10(x*k) + b

def SecondOrder(x, a, k, b):
    return a*np.log10(k*x**.25) + b

fig = plt.figure()

data = np.genfromtxt("M81R_BStackWCSPlotValues.tsv", delimiter="\t",

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])


mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(x,y)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(Second
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("de Vaucouleurs on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(0,70)
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.

plt.subplot(2, 2, 1)
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(10000,70000)
plt.title('M81 R-B de Vaucouleurs')
```

```
plt.title('M87 R-B de Vaucouleurs')
plt.xlabel('Pixels')
plt.ylabel('Counts')
#plt.show()

data = np.genfromtxt("M87R_BStackWCSPlotValues.tsv", delimiter="\t",

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])

#x, y = xy.T
mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(x,y)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(Second
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("de Vaucouleurs on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(0,70)
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.

plt.subplot(2, 2, 2)
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(10000,70000)
plt.title('M87 R-B de Vaucouleurs')
plt.xlabel('Pixels')
#plt.ylabel('Counts')
plt.show()

plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
plt.ylim(0,4000)
plt.title('M87 R-B de Vaucouleurs')
plt.xlabel('Pixels')
plt.ylabel('Counts')
plt.savefig("M87R_BdeVaucouleurs.png")
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 galazy to the brightest star in the Blue filter:

In [ ]:

```
Elipse

center=148.88831 69.066062
fk5
1 pixel = 0.59893708 arcsec

reg sum     error   area        surf_bri        surf_err
                (arcsec**2)     (sum/arcsec**2) (sum/arcsec**2)
--- ---     -----   ----------      --------------- ---------------
1   69406464        8331.05 10031.8     6918.67     0.830468

reg sum npix    mean    median  min max var stddev  rms
--- --- ----    ----    ------  --- --- --- ------  ---
1   69406464    27965   2481.9  2401    1375    7930    139577  373.

Cirle Mag 11.338

{148.937561500 +69.029386700 796-020040 148.9375615 +69.0293867 19 1

center=148.93874 69.029964
fk5
1 pixel = 0.59893708 arcsec

reg sum     error   area        surf_bri        surf_err
                (arcsec**2)     (sum/arcsec**2) (sum/arcsec**2)
--- ---     -----   ----------      --------------- ---------------
1   2026251     1423.46 182.591     11097.2     7.7959

reg sum npix    mean    median  min max var stddev  rms
--- --- ----    ----    ------  --- --- --- ------  ---
1   2026251 509 3980.85 2607    2050    29016   1.39178e+07 3730.65
```

M81 Red

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

In [ ]:

```
Elipse

center=148.88831 69.066062
fk5
1 pixel = 0.59830185 arcsec

reg sum     error   area        surf_bri        surf_err
                (arcsec**2)     (sum/arcsec**2) (sum/arcsec**2)
--- ---     -----   ----------      --------------- ---------------
1   76651327        8755.07 10032.7     7640.16     0.872655

reg sum npix    mean    median  min max var stddev  rms
--- --- ----    ----    ------  --- --- --- ------  ---
1   76651327    28027   2734.91 2543    0   22279   619839  787.299

Circle 11.338

{148.937561500 +69.029386700 796-020040 148.9375615 +69.0293867 19 1

center=148.93729 69.029608
fk5
1 pixel = 0.59830185 arcsec

reg sum     error   area        surf_bri        surf_err
                (arcsec**2)     (sum/arcsec**2) (sum/arcsec**2)
--- ---     -----   ----------      --------------- ---------------
1   1526263     1235.42 94.8608     16089.5     13.0235

reg sum npix    mean    median  min max var stddev  rms
--- --- ----    ----    ------  --- --- --- ------  ---
1   1526263 265 5759.48 2898    2217    49463   5.55042e+07 7450.11
```

M87 Blue

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

In [ ]:

```
Galaxy

center=187.70621 12.390445
fk5
1 pixel = 0.59883966 arcsec

reg sum     error   area        surf_bri        surf_err
                (arcsec**2)     (sum/arcsec**2) (sum/arcsec**2)
--- ---     -----   ----------      -------------- --------------
1   11496322        3390.62 1489.66     7717.41     2.2761

reg sum npix    mean    median  min max var stddev  rms
--- --- ----    ----    ------  --- --- --- ------  ---
1   11496322    4154    2767.53 2737    2128    3580    35543.4 188.5

Star 8.860

{187.703899200 +12.486724800 513-054652 187.7038992 +12.4867248 9 8.8

center=187.70373 12.486656
fk5
1 pixel = 0.59883966 arcsec

reg sum     error   area        surf_bri        surf_err
                (arcsec**2)     (sum/arcsec**2) (sum/arcsec**2)
--- ---     -----   ----------      -------------- --------------
1   12774613        3574.16 654.82      19508.6     5.45823

reg sum npix    mean    median  min max var stddev  rms
--- --- ----    ----    ------  --- --- --- ------  ---
1   12774613    1826    6995.95 3290    2435    64144   1.12761e+08 1
```

M87 Red

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

In [ ]:

```
Galaxy

center=187.70621 12.390445
fk5
1 pixel = 0.59882223 arcsec

reg sum     error   area        surf_bri        surf_err
                (arcsec**2)     (sum/arcsec**2) (sum/arcsec**2)
--- ---     -----   ----------      -------------- --------------
1   15572807        3946.24 1489.93     10452       2.6486

reg sum npix    mean    median  min max var stddev  rms
--- --- ----    ----    ------  --- --- --- ------  ---
1   15572807    4155    3747.97 3655    2884    5342    122046  349.

Star 8.860

{187.703899200 +12.486724800 513-054652 187.7038992 +12.4867248 9 8.

center=187.70373 12.486656
fk5
1 pixel = 0.59882223 arcsec

reg sum     error   area        surf_bri        surf_err
                (arcsec**2)     (sum/arcsec**2) (sum/arcsec**2)
--- ---     -----   ----------      -------------- --------------
1   16062810        4007.84 654.065     24558.4     6.1276

reg sum npix    mean    median  min max var stddev  rms
--- --- ----    ----    ------  --- --- --- ------  ---
1   16062810    1824    8806.37 4278    3114    65273   1.63528e+08
```

Supporting Analysis

M87 Analysis - Eliptical Galaxy Radial Profile Curve Fitting - Exponential Curve

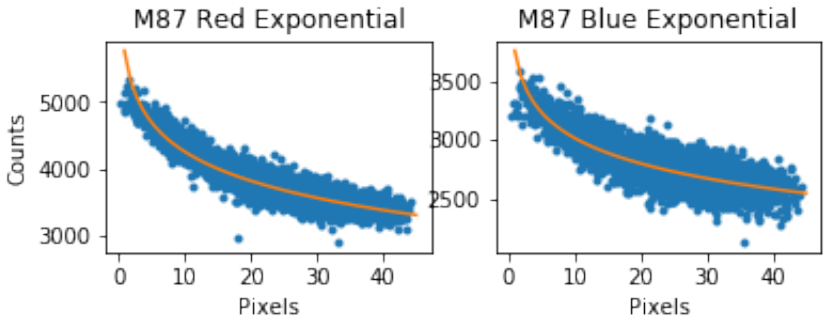
In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as scypy
```

```
from astropy.stats import sigma_clipped_stats
from scipy import stats
import warnings
warnings.filterwarnings('ignore')


def exponential(x, a, k, b):
    return a*np.log10(x*k) + b

#def SecondOrder(x, a, k, b):
#    return a*np.log10(k*x**.25) + b

fig = plt.figure()

data = np.genfromtxt("M87RedPlotValues.tsv", delimiter="\t", skip_he

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])


mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(x,y)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(expone
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("Exponential on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(0,45)
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +

plt.subplot(2, 2, 1)
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(10000,70000)
plt.title('M87 Red Exponential')
plt.xlabel('Pixels')
plt.ylabel('Counts')
#plt.show()

data = np.genfromtxt("M87BluePlotValues.tsv", delimiter="\t", skip_h

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])

```

```
#x, y = xy.T
mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(x,y)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(expone
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("Exponential on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(0,45)
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +

plt.subplot(2, 2, 2)
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(10000,70000)
plt.title('M87 Blue Exponential')
plt.xlabel('Pixels')
#plt.ylabel('Counts')
plt.savefig("M87exponential.png")
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
```



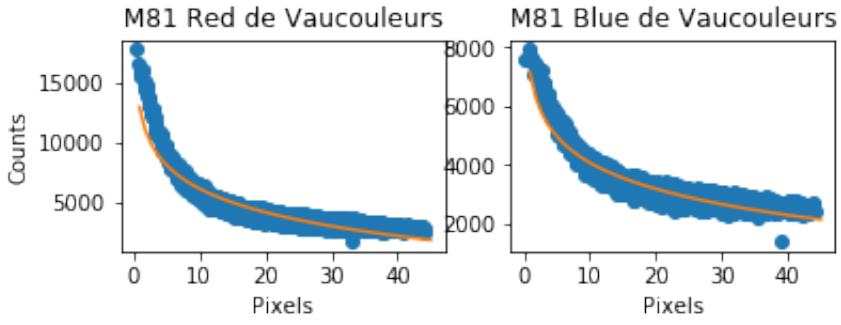
M81 Analysis - Spiral Galaxy Radial Profile Curve Fitting - de Vaucouleurs Curve

In [5]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as scypy
from astropy.stats import sigma_clipped_stats
from scipy import stats

#def exponential(x, a, k, b):
#    return a*np.log10(x*k) + b

def SecondOrder(x, a, k, b):
    return a*np.log10(k*x**.25) + b

fig = plt.figure()

data = np.genfromtxt("M81RedPlotValues.tsv", delimiter="\t", skip_he

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])


mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(x,y)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(Second
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("de Vaucouleurs on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(0,45)
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**.

plt.subplot(2, 2, 1)
plt.plot(x, y, linestyle = 'None', marker = 'o')
plt.plot(xtest,testy)
#plt.ylim(10000,70000)
plt.title('M81 Red de Vaucouleurs')
plt.xlabel('Pixels')
plt.ylabel('Counts')
#plt.show()

data = np.genfromtxt("M81BluePlotValues.tsv", delimiter="\t", skip_h

#x, y = xy.T
x = (data[:,0])
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 = scypy.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_expone
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 - Risidual Analysis - Radial Profile Curve Fitting - Exponential Curve

In [6]:

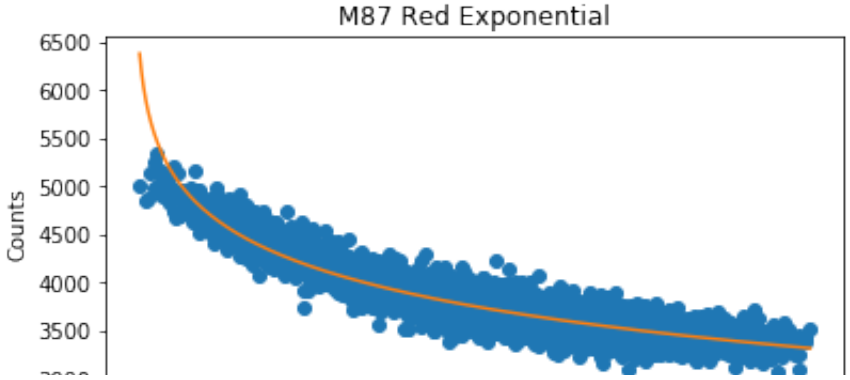
```
import numpy as np
import matplotlib.pyplot as plt
import scipy as scypy
from astropy.stats import sigma_clipped_stats
from scipy import stats
import warnings
warnings.filterwarnings('ignore')

def exponential(x, a, k, b):
    return a*np.log10(x*k) + b

fig = plt.figure()

data = np.genfromtxt("M87RedPlotValues.tsv", delimiter="\t", skip_he

x = (data[:,0])
y = (data[:,1])

mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(x,y)
#print(min(x),max(x))
#print(x.shape)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(expone
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("Exponential on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(min(x),max(x),3969)
testy = (popt_exponential[0]*np.log10(xtest*popt_exponential[1]) + p
diffy = (popt_exponential[0]*np.log10(x*popt_exponential[1]) + popt_

#plt.subplot(2, 2, 1)
frame1=fig.add_axes((.1,.3,.8,.6))
plt.plot(x, y, linestyle = 'None', marker = 'o')
plt.plot(xtest,testy)
#plt.ylim(12000,18000)
plt.ylabel('Counts')
plt.title('M87 Red Exponential')


#Residual plot
difference = y - diffy
frame2=fig.add_axes((.1,.1,.8,.2))
plt.plot(x,difference,'or')

```

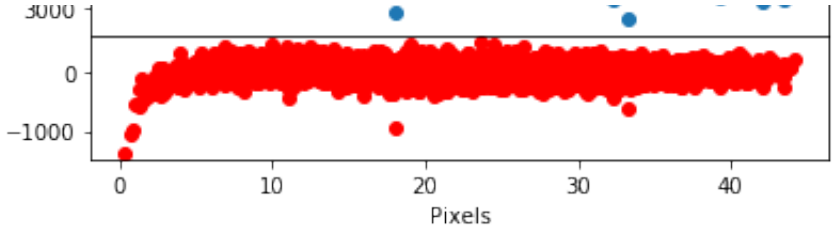
```
plt.xlabel('Pixels')
plt.show()

fig = plt.figure()

data = np.genfromtxt("M87BluePlotValues.tsv", delimiter="\t", skip_h

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])

#x, y = xy.T
mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(min(x),max(x))
#print(x,y)
#print(x.shape)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(expone
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("Exponential on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(min(x),max(x),3969)
testy = (popt_exponential[0]*np.log10(xtest*popt_exponential[1]) + p
diffy = (popt_exponential[0]*np.log10(x*popt_exponential[1]) + popt_

#plt.subplot(2, 2, 2)
frame1=fig.add_axes((.1,.3,.8,.6))
plt.plot(x, y, linestyle = 'None', marker = 'o')
plt.plot(xtest,testy)
#plt.ylim(12000,18000)
plt.ylabel('Counts')
plt.title('M87 Blue Exponential')


#Residual plot
difference = y - diffy
frame2=fig.add_axes((.1,.1,.8,.2))
plt.plot(x,difference,'or')

plt.xlabel('Pixels')
plt.show()

fig = plt.figure()

data = np.genfromtxt("M81RedPlotValues.tsv", delimiter="\t", skip_he
```

```
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 = scypy.optimize.curve_fit(expone
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_expone
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_h
146
147 #x, y = xy.T
148 x = (data[:,0])
149 y = (data[:,1])
150
151 mean, median, std = sigma_clipped_stats(y)
```

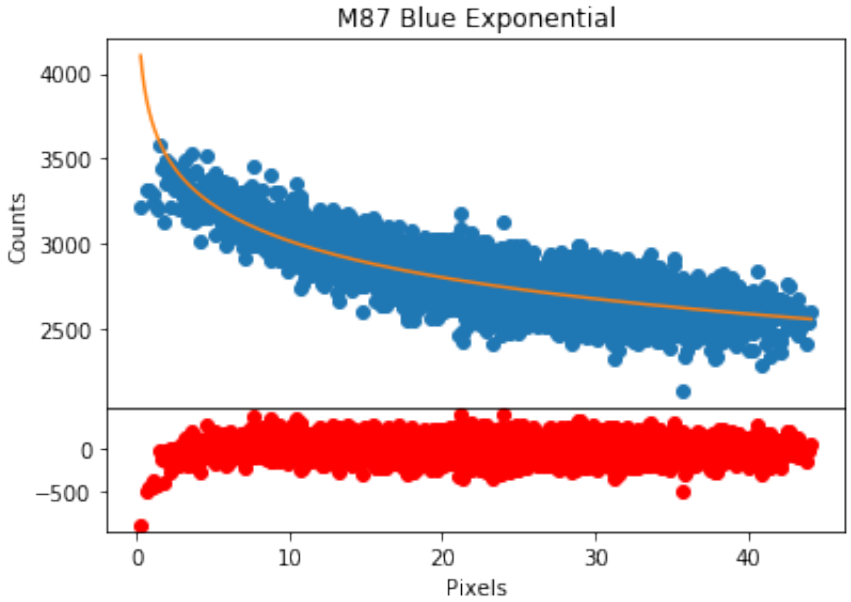
```
#print(min(y),max(y),mean,median,std)
#print(min(x),max(x))
#print(x,y)
#print(x.shape)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(expone
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("Exponential on the rax data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(min(x),max(x),3969)
testy = (popt_exponential[0]*np.log10(xtest*popt_exponential[1]) + p
diffy = (popt_exponential[0]*np.log10(x*popt_exponential[1]) + popt_

#plt.subplot(2, 2, 4)
frame1=fig.add_axes((.1,.3,.8,.6))
plt.plot(x, y, linestyle = 'None', marker = 'o')
plt.plot(xtest,testy)
#plt.ylim(15000,25000)
plt.ylabel('Counts')
plt.title('M81 Blue Exponential')


#Residual plot
difference = y - diffy
frame2=fig.add_axes((.1,.1,.8,.2))
plt.plot(x,difference,'or')

plt.xlabel('Pixels')
#plt.set_yticklabels([])
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
```

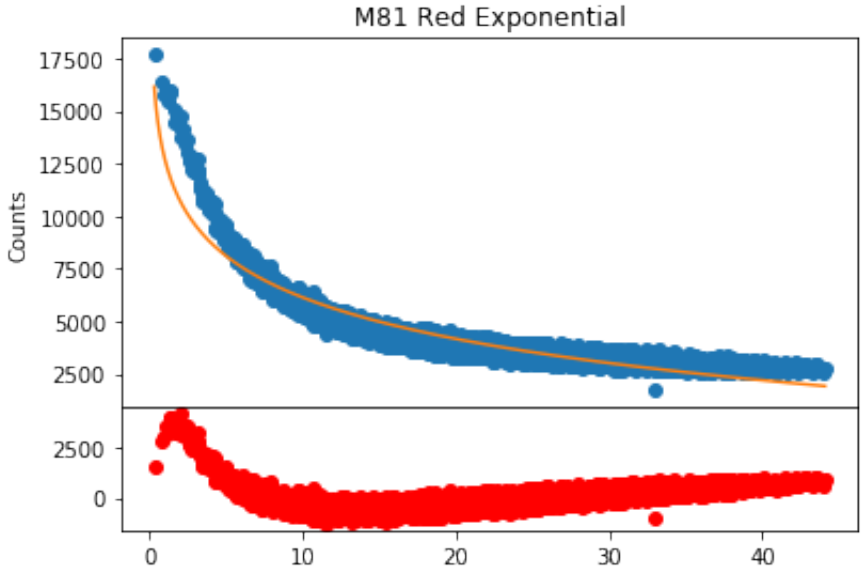





```
Exponential on the raw data
pre-exponential factor = -714.8465042582 (+/-) 7.4201598756
rate constant = 986.1767 (+/-) 68607035.7887
yintercept = 5870.2950 (+/-) 21597873.6918
```

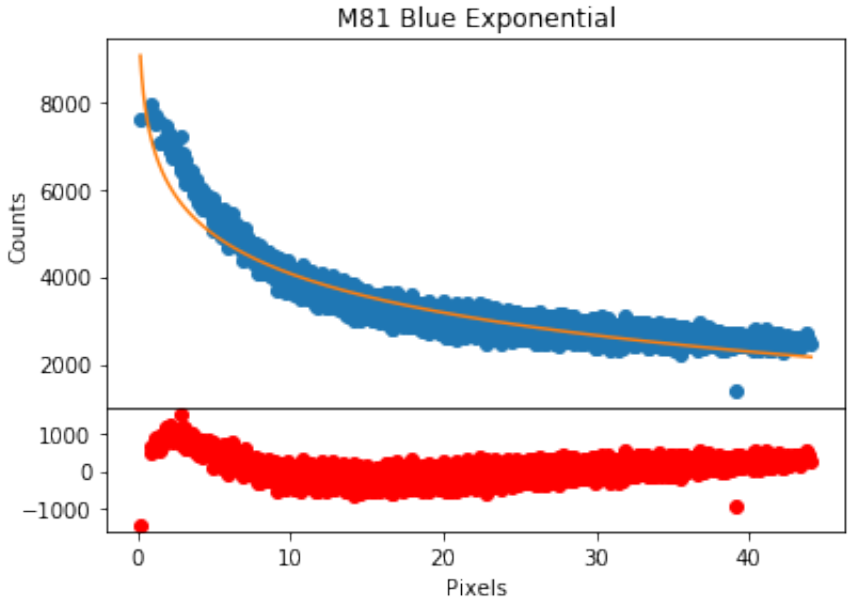



```
Exponential on the rax data
pre-exponential factor = -6527.2206979377 (+/-) 39.8917972998
rate constant = 2.3298 (+/-) 393289.5814
yintercept = 15062.6515 (+/-) 477989226.6355
```



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 - Risidual Analysis - Radial Profile Curve Fitting - de Vaucouleurs Curve

In [12]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as scypy
from astropy.stats import sigma_clipped_stats
from scipy import stats

#def exponential(x, a, k, b):
#    return a*np.log10(x*k) + b
def SecondOrder(x, a, k, b):
    return a*np.log10(k*x**.25) + b


fig = plt.figure()

data = np.genfromtxt("M87RedPlotValues.tsv", delimiter="\t", skip_he

x = (data[:,0])
y = (data[:,1])

mean, median, std = sigma_clipped_stats(y)
```

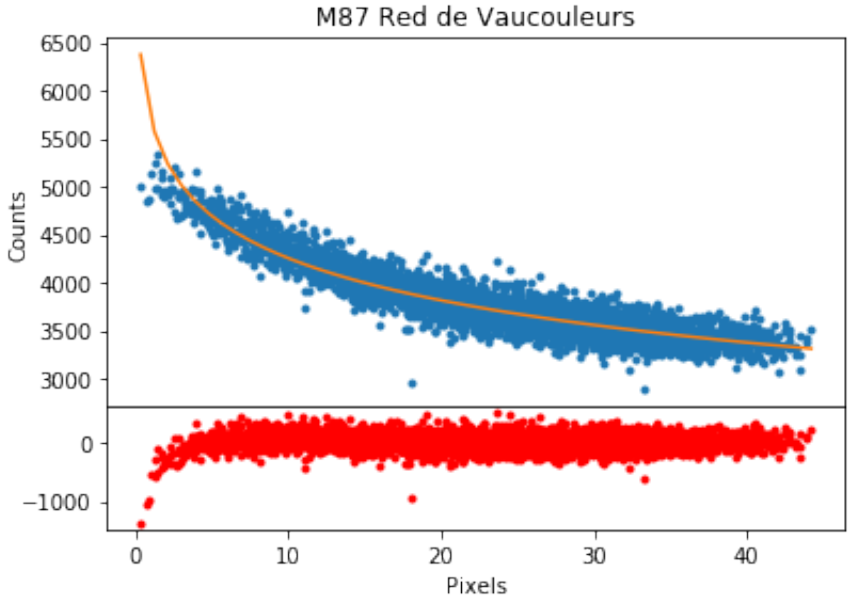
```
#print(min(y),max(y),mean,median,std)
#print(x,y)
#print(min(x),max(x))
#print(x.shape)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(Second
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("de Vaucouleurs on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(min(x),max(x))
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**
diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x**.25)

#plt.subplot(2, 2, 1)
frame1=fig.add_axes((.1,.3,.8,.6))
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(12000,18000)
plt.ylabel('Counts')
plt.title('M87 Red de Vaucouleurs')

#Residual plot
difference = y - diffy
frame2=fig.add_axes((.1,.1,.8,.2))
plt.plot(x,difference,'or', marker = '.')

plt.xlabel('Pixels')
plt.show()

fig = plt.figure()

data = np.genfromtxt("M87BluePlotValues.tsv", delimiter="\t", skip_h

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])

#x, y = xy.T
mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(min(x),max(x))
#print(x,y)
#print(x.shape)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(Second
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("de Vaucouleurs on the raw data")
```

```
print ( de Vaucouleurs on the rax data )
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(min(x),max(x))
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**
diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x**.25)

#plt.subplot(2, 2, 2)
frame1=fig.add_axes((.1,.3,.8,.6))
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(12000,18000)
plt.ylabel('Counts')
plt.title('M87 Blue de Vaucouleurs')

#Residual plot
difference = y - diffy
frame2=fig.add_axes((.1,.1,.8,.2))
plt.plot(x,difference,'or', marker = '.')

plt.xlabel('Pixels')
plt.show()

fig = plt.figure()

data = np.genfromtxt("M81RedPlotValues.tsv", delimiter="\t", skip_he

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])

#x, y = xy.T
mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(min(x),max(x))
#print(x,y)
#print(x.shape)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(Second
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("de Vaucouleurs on the rax data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(min(x),max(x))
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**
```

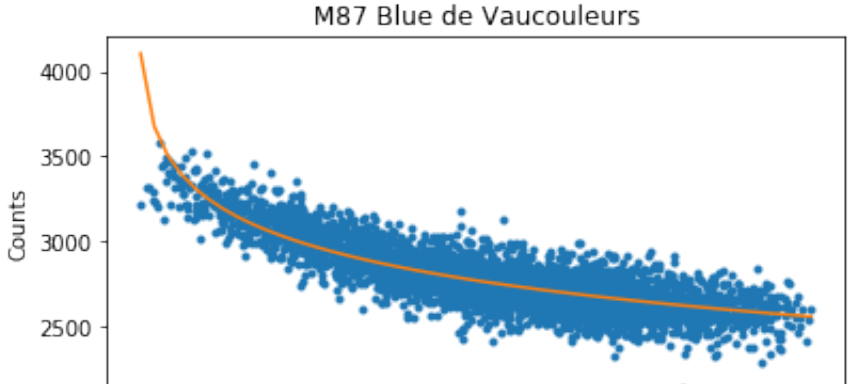
```
diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x**.25)

#plt.subplot(2, 2, 3)
frame1=fig.add_axes((.1,.3,.8,.6))
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(15000,25000)
plt.ylabel('Counts')
plt.title('M81 Red de Vaucouleurs')

#Residual plot
difference = y - diffy
frame2=fig.add_axes((.1,.1,.8,.2))
plt.plot(x,difference,'or', marker = '.')

plt.xlabel('Pixels')
plt.show()


fig = plt.figure()

data = np.genfromtxt("M81BluePlotValues.tsv", delimiter="\t", skip_h

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])

mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(min(x),max(x))
#print(x,y)
#print(x.shape)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(Second
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("de Vaucouleurs on the rax data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(min(x),max(x))
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**
diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x**.25)

#plt.subplot(2, 2, 4)
frame1=fig.add_axes((.1,.3,.8,.6))
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(15000,25000)
plt.ylabel('Counts')
```

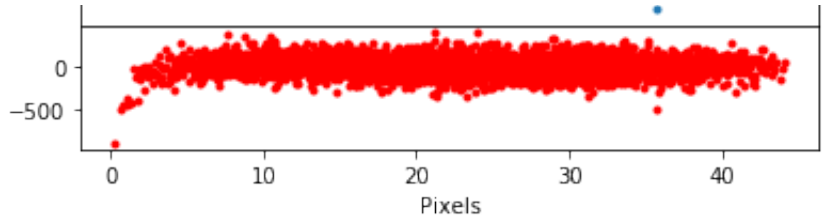
```
172 plt.ylabel('Counts')
173 plt.title('M81 Blue de Vaucouleurs')
174
175
176 #Residual plot
177 difference = y - diffy
178 frame2=fig.add_axes((.1,.1,.8,.2))
179 plt.plot(x,difference,'or', marker = '.')
180
181 plt.xlabel('Pixels')
182 #plt.set_yticklabels([])
183 plt.show()
```

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

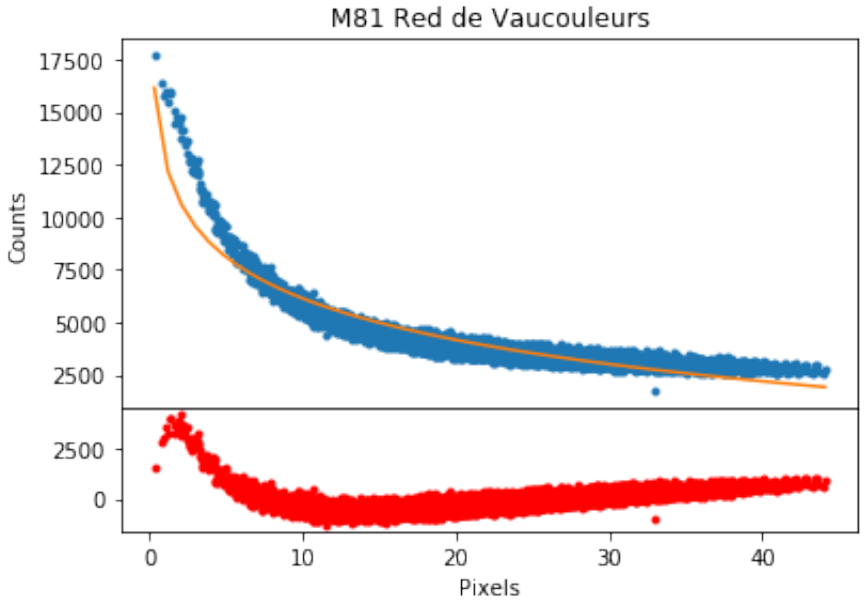



```
de Vaucouleurs on the raw data
pre-exponential factor = -2859.3860296192 (+/-) 29.6808886044
rate constant = 45.3364 (+/-) 1777622.6717
yintercept = 8466.4980 (+/-) 48691125.9814
```

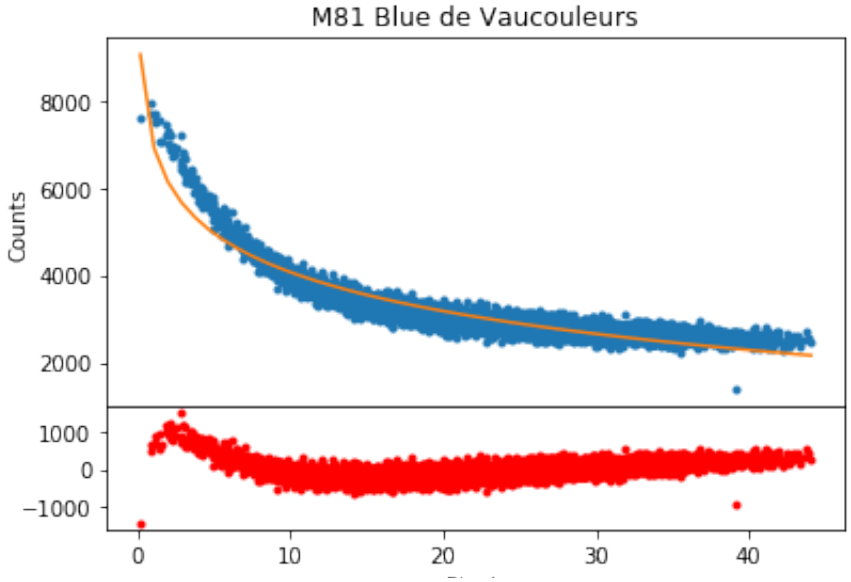





```
de Vaucouleurs on the rax data
pre-exponential factor = -26108.8827771398 (+/-) 159.5723836325
rate constant = 3.7826 (+/-) 195288.7901
yintercept = 27750.4116 (+/-) 585414854.0087
```




```
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 - Risidual Analysis - Radial Profile Curve Fitting - R-B Filters

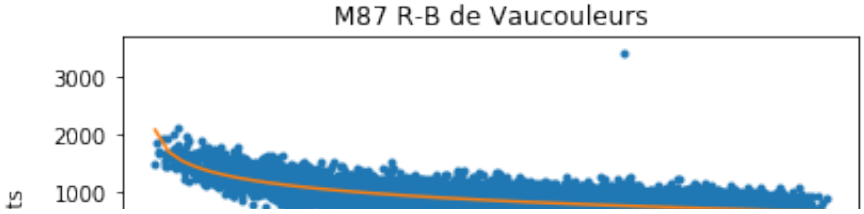
In [15]:

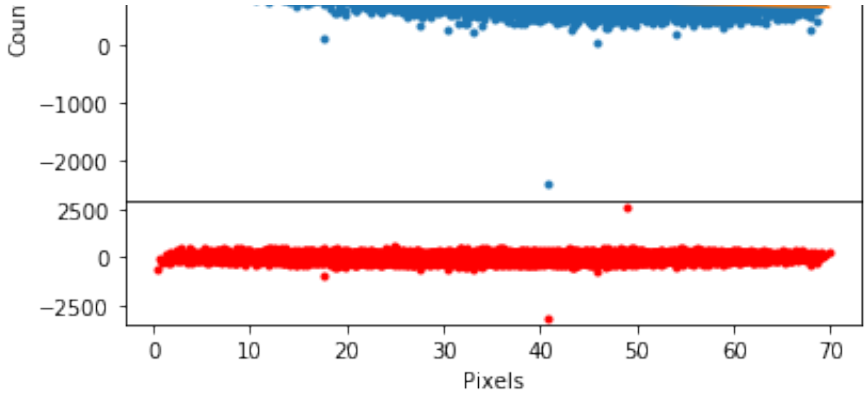
```
import numpy as np
import matplotlib.pyplot as plt
import scipy as scypy
from astropy.stats import sigma_clipped_stats
from scipy import stats

def exponential(x, a, k, b):
    return a*np.log10(x*k) + b
def SecondOrder(x, a, k, b):
    return a*np.log10(k*x**.25) + b


fig = plt.figure()

data = np.genfromtxt("M87R_BStackWCSPlotValues.tsv", delimiter="\t",

x = (data[:,0])
y = (data[:,1])

mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(x,y)
#print(min(x),max(x))
#print(x.shape)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(Second
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("de Vaucouleurs on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(min(x),max(x))
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**
diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x**.25)

#plt.subplot(2, 2, 1)
frame1=fig.add_axes((.1,.3,.8,.6))
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(12000,18000)
plt.ylabel('Counts')
```

```
plt.title('M87 R-B de Vaucouleurs')


#Residual plot
difference = y - diffy
frame2=fig.add_axes((.1,.1,.8,.2))
plt.plot(x,difference,'or', marker = '.')

plt.xlabel('Pixels')
plt.show()

fig = plt.figure()

data = np.genfromtxt("M87R_BStackWCSPlotValues.tsv", delimiter="\t",

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])

#x, y = xy.T
mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(min(x),max(x))
#print(x,y)
#print(x.shape)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(expone
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("Exponential on the raw data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(min(x),max(x))
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest) +
diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x) + pop

#plt.subplot(2, 2, 2)
frame1=fig.add_axes((.1,.3,.8,.6))
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(12000,18000)
plt.ylabel('Counts')
plt.title('M87 R-B Exponential')


#Residual plot
difference = y - diffy
frame2=fig.add_axes((.1,.1,.8,.2))
plt.plot(x,difference,'or', marker = '.')
```

```
plt.plot(x,difference, 'or' , marker = '.' )

plt.xlabel('Pixels')
plt.show()

fig = plt.figure()

data = np.genfromtxt("M81R_BStackWCSPlotValues.tsv", delimiter="\t",

#x, y = xy.T
x = (data[:,0])
y = (data[:,1])

#x, y = xy.T
mean, median, std = sigma_clipped_stats(y)
#print(min(y),max(y),mean,median,std)
#print(min(x),max(x))
#print(x,y)
#print(x.shape)

popt_exponential, pcov_exponential = scypy.optimize.curve_fit(Second
perr_exponential = np.sqrt(np.diag(pcov_exponential))
print ("de Vaucouleurs on the rax data")
#print (popt_exponential, pcov_exponential)
print ("pre-exponential factor = %0.10f (+/-) %0.10f" % (popt_expone
print ("rate constant = %0.4f (+/-) %0.4f" %(popt_exponential[1], pe
print ("yintercept = %0.4f (+/-) %0.4f" %(popt_exponential[2], perr_

xtest = np.linspace(min(x),max(x))
testy = (popt_exponential[0]*np.log10((popt_exponential[1])*xtest**
diffy = (popt_exponential[0]*np.log10((popt_exponential[1])*x**.25)

#plt.subplot(2, 2, 3)
frame1=fig.add_axes((.1,.3,.8,.6))
plt.plot(x, y, linestyle = 'None', marker = '.')
plt.plot(xtest,testy)
#plt.ylim(15000,25000)
plt.ylabel('Counts')
plt.title('M81 R-B de Vaucouleurs')


#Residual plot
difference = y - diffy
frame2=fig.add_axes((.1,.1,.8,.2))
plt.plot(x,difference,'or', marker = '.')

plt.xlabel('Pixels')
plt.show()


fig = plt.figure()
```

```
145
146 data = np.genfromtxt("M81R_BStackWCSPlotValues.tsv", delimiter="\t"
147 #data = np.genfromtxt("M81RedPlotValues.tsv", delimiter="\t", skip_
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 = scypy.optimize.curve_fit(expone
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_expone
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). AstroImageJ: 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