

Sigma Background estimation with 4 boxes, 40x40 in size

December 6, 2018

```
In [213]: import astropy
import matplotlib.pyplot as plt
from astropy.io import fits
import numpy as np
from scipy.stats import norm
import matplotlib.mlab as mlab
import glob
from scipy import ndimage
from scipy.optimize import curve_fit
from astropy.modeling.models import Linear1D

hdu_list = fits.open('atlas_pleione.fit')
hdu_list.info()
```

Filename: atlas_pleione.fit

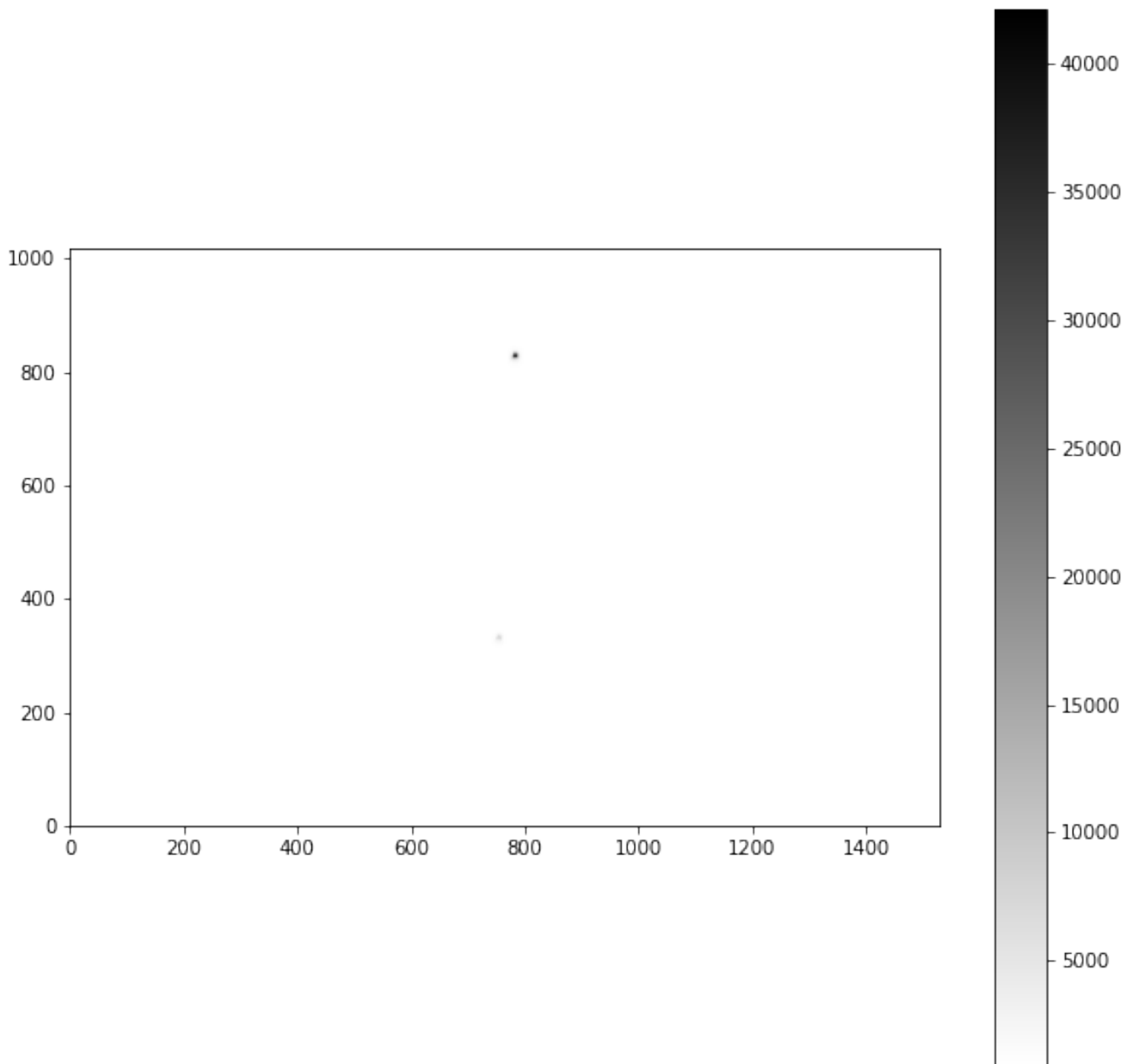
| No. | Name | Ver | Type | Cards | Dimensions | Format |
|-----|---------|-----|------------|-------|--------------|----------------------------|
| 0 | PRIMARY | 1 | PrimaryHDU | 38 | (1530, 1020) | int16 (rescales to uint16) |

```
In [214]: image_data = hdu_list[0].data ## 2.3 #ADU multiplied by the gain gives number of p.e.
print(image_data)
print(image_data.shape)
#rows 1020, columns 1530
```

```
[[949 941 937 ... 960 945 960]
 [937 936 944 ... 952 949 948]
 [938 944 930 ... 950 948 952]
 ...
 [953 947 936 ... 954 954 961]
 [935 942 948 ... 967 952 947]
 [945 930 935 ... 953 964 958]]
(1020, 1530)
```

```
In [215]: plt.figure(figsize=(10,10))
plt.imshow(image_data, cmap='gray_r', origin='lower')
plt.colorbar()
```

Out [215]: <matplotlib.colorbar.Colorbar at 0x1a2abb7b38>



```
In [216]: #take 4 boxes at the corners of size 40by40
#first argument: row, second: column
boxUL = image_data[0:40,0:40]
print(boxUL.shape)
print('median UL:', np.median(boxUL))
boxDL = image_data[980:1020,0:40]
```

```

print(boxDL.shape)
print('median DL:', np.median(boxDL))
boxUR = image_data[0:40,1490:1530]
print(boxUR.shape)
print('median UR:', np.median(boxUR))
boxDR = image_data[980:1020,1490:1530]
print(boxDR.shape)
print('median DR:', np.median(boxDR))

l = [np.median(boxUL), np.median(boxDL), np.median(boxUR), np.median(boxDR)]
print('\n', 'average of pixel value in boxes is the sigma of background:', np.mean(l))

```

```

(40, 40)
median UL: 937.0
(40, 40)
median DL: 937.0
(40, 40)
median UR: 949.0
(40, 40)
median DR: 949.0

```

average of pixel value in boxes is the sigma of background: 943.0

```

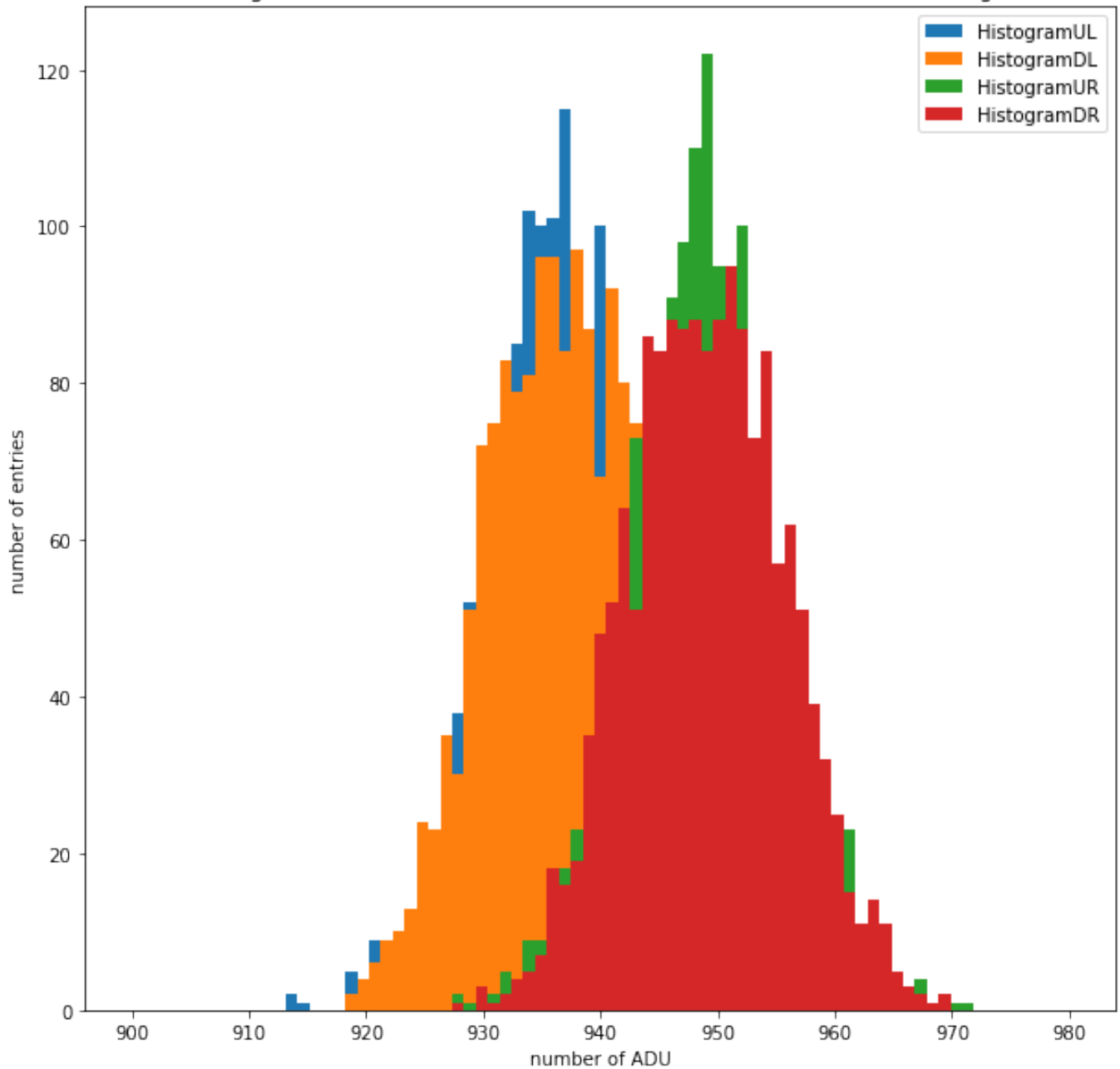
In [217]: plt.figure(figsize=(10,10))
bins = np.linspace(900, 980, 80)
plt.hist(boxUL.flatten(), bins=bins, label='HistogramUL')
plt.hist(boxDL.flatten(), bins=bins, label='HistogramDL')
plt.hist(boxUR.flatten(), bins=bins, label='HistogramUR')
plt.hist(boxDR.flatten(), bins=bins, label='HistogramDR')

plt.title('histograms of 4 boxes with size 40x40 taken at the corners of the image')
plt.xlabel('number of ADU')
plt.ylabel('number of entries')
plt.legend()

```

Out[217]: <matplotlib.legend.Legend at 0x1a2a2c9c50>

histograms of 4 boxes with size 40x40 taken at the corners of the image



```
In [218]: plt.figure(figsize=(10,10))
```

```
def fit_function(x, A, mu, sigma):  
    return (A * np.exp(-1.0 * (x - mu)**2 / (2 * sigma**2)))
```

```
# Histograms
```

```
hdu_list = fits.open('atlas_pleione.fit')
```

```

imgdata = hdu_list[0].data

boxUL = image_data[0:40,0:40]

data = boxUL.flatten() #return 1D array
bins = np.linspace(900, 980, 80)

data_entries, bins = np.histogram(data, bins=bins)

binscenters = np.array([0.5 * (bins[i] + bins[i+1]) for i in range(len(bins)-1)])

# Fit the function to the histogram data.
popt, pcov = curve_fit(fit_function, xdata=binscenters, ydata=data_entries, p0=[200, 930, 6])
print(popt)

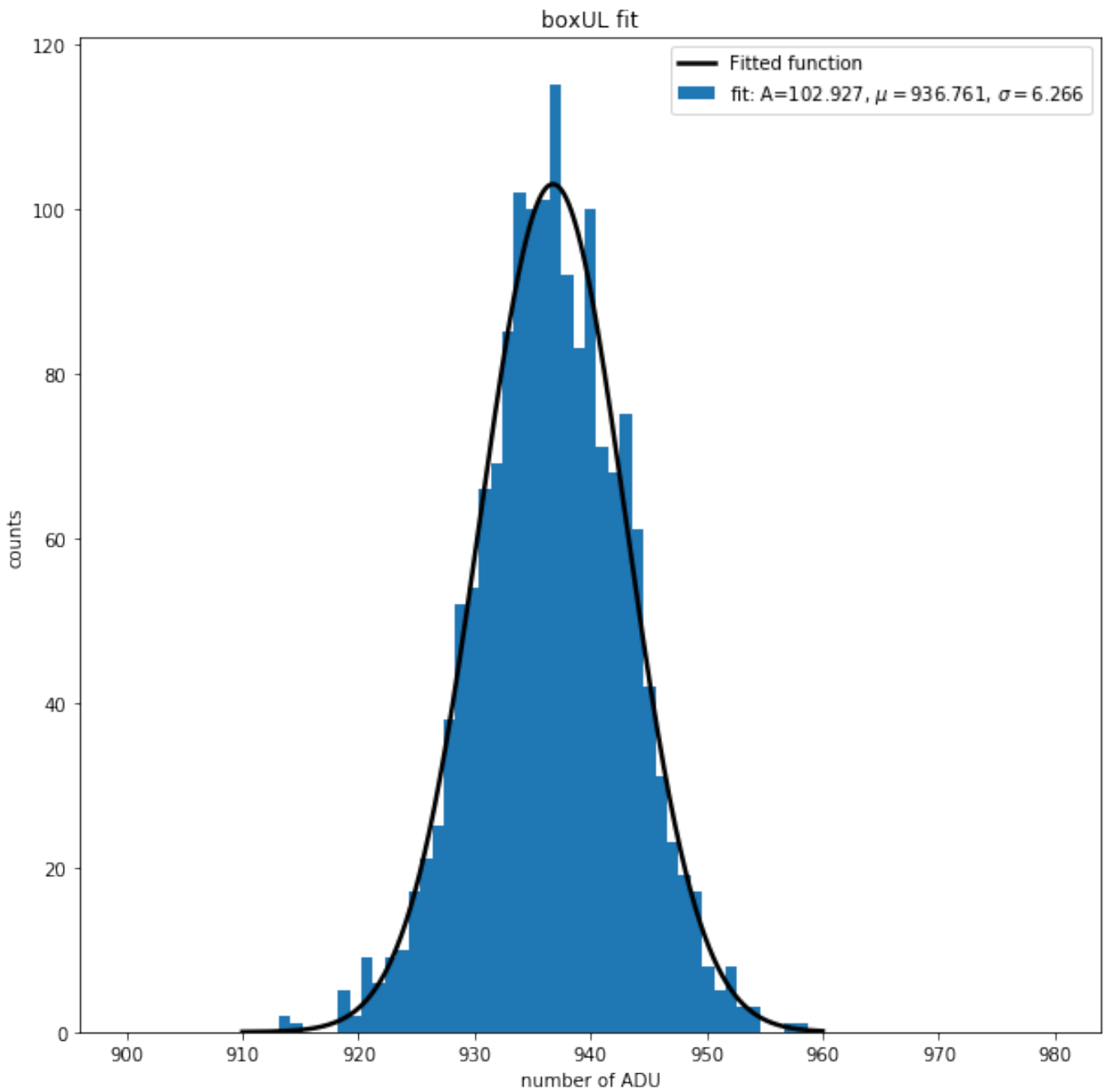
# Generate enough x values to make the curves look smooth.
xspace = np.linspace(910, 960, 10000)

# Plot the histogram and the fitted function.
plt.bar(binscenters, data_entries, width=bins[1] - bins[0], label='fit: A=%5.3f, $\mu$=%5.3f$',
plt.plot(xspace, fit_function(xspace, *popt), color='k', linewidth=2.5, label=r'Fitted function')
plt.title("boxUL fit")
plt.xlabel("number of ADU")
plt.ylabel("counts")
plt.legend(loc='best')
plt.show()

poptUL = tuple(popt)

```

```
[102.92667463 936.76050701 6.26587862]
```



```
In [219]: plt.figure(figsize=(10,10))
```

```
def fit_function(x, A, mu, sigma):  
    return (A * np.exp(-1.0 * (x - mu)**2 / (2 * sigma**2)))
```

```

hdu_list = fits.open('atlas_pleione.fit')
imgdata = hdu_list[0].data

boxDL = image_data[980:1020,0:40]

data = boxDL.flatten() #return 1D array

bins = np.linspace(900, 980, 80)

data_entries, bins = np.histogram(data, bins=bins)

binscenters = np.array([0.5 * (bins[i] + bins[i+1]) for i in range(len(bins)-1)])

# Fit the function to the histogram data.
popt, pcov = curve_fit(fit_function, xdata=binscenters, ydata=data_entries, p0=[175, 930, 6])
print(popt)

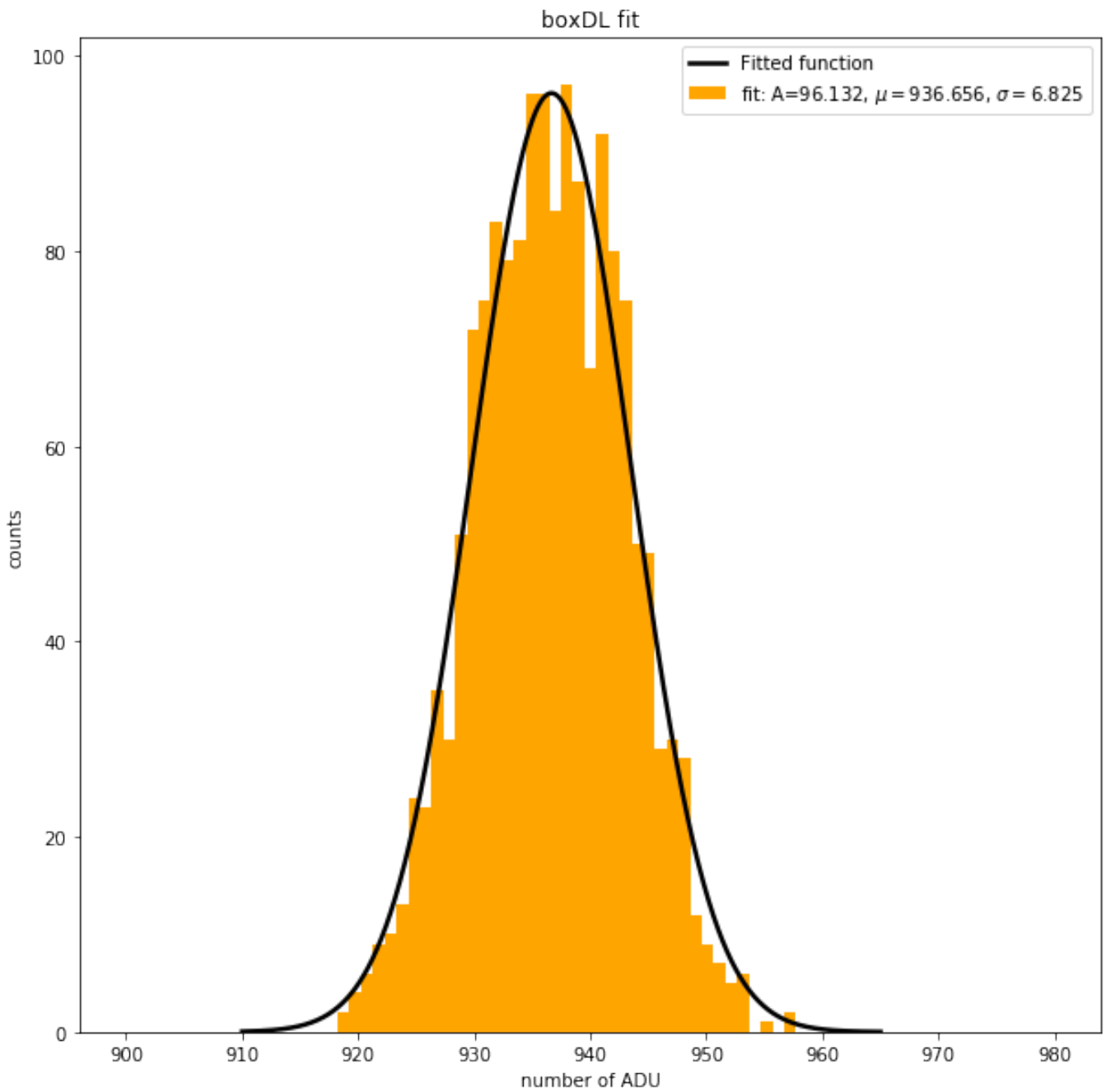
# Generate enough x values to make the curves look smooth.
xspace = np.linspace(910, 965, 10000)

# Plot the histogram and the fitted function.
plt.bar(binscenters, data_entries, width=bins[1] - bins[0], color='orange', label='fit: A=%5.1f')
plt.plot(xspace, fit_function(xspace, *popt), color='k', linewidth=2.5, label=r'Fitted function')
plt.title("boxDL fit")
plt.xlabel("number of ADU")
plt.ylabel("counts")
plt.legend(loc='best')
plt.show()

poptDL = tuple(popt)

```

```
[ 96.13220564  936.65644429   6.82539114]
```



```
In [220]: plt.figure(figsize=(10,10))
```

```
def fit_function(x, A, mu, sigma):  
    return (A * np.exp(-1.0 * (x - mu)**2 / (2 * sigma**2)))
```



```

hdu_list = fits.open('atlas_pleione.fit')
imgdata = hdu_list[0].data

boxUR = image_data[0:40,1490:1530]

data = boxUR.flatten() #return 1D array

bins = np.linspace(900, 980, 80)
data_entries, bins = np.histogram(data, bins=bins)

binscenters = np.array([0.5 * (bins[i] + bins[i+1]) for i in range(len(bins)-1)])

# Fit the function to the histogram data.
popt, pcov = curve_fit(fit_function, xdata=binscenters, ydata=data_entries, p0=[200, 930, 6])
print(popt)

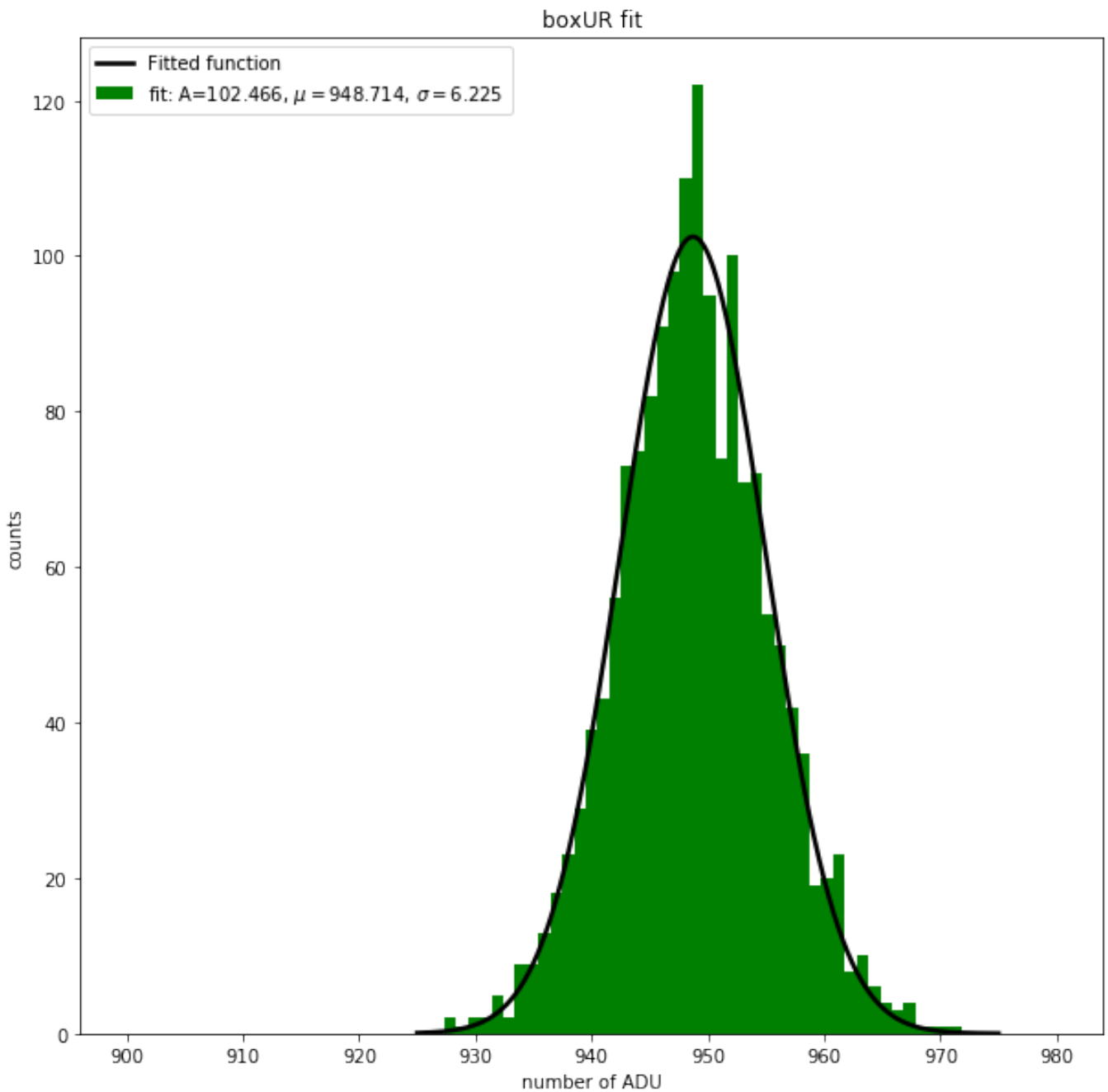
# Generate enough x values to make the curves look smooth.
xspace = np.linspace(925, 975, 10000)

# Plot the histogram and the fitted function.
plt.bar(binscenters, data_entries, width=bins[1] - bins[0], color='green', label='fit: A=%5.3f')
plt.plot(xspace, fit_function(xspace, *popt), color='k', linewidth=2.5, label=r'Fitted function')
plt.title("boxUR fit")
plt.xlabel("number of ADU")
plt.ylabel("counts")
plt.legend(loc='best')
plt.show()

poptUR = tuple(popt)

```

```
[102.46621147 948.71394493 6.22540316]
```



```
In [221]: plt.figure(figsize=(10,10))
```

```
def fit_function(x, A, mu, sigma):
    return (A * np.exp(-1.0 * (x - mu)**2 / (2 * sigma**2)))
```

```
hdu_list = fits.open('atlas_pleione.fit')
imgdata = hdu_list[0].data
```

```

boxDR = image_data[980:1020,1490:1530]

data = boxDR.flatten() #return 1D array

bins = np.linspace(900, 980, 80)
data_entries, bins = np.histogram(data, bins=bins)

binscenters = np.array([0.5 * (bins[i] + bins[i+1]) for i in range(len(bins)-1)])

# Fit the function to the histogram data.
popt, pcov = curve_fit(fit_function, xdata=binscenters, ydata=data_entries, p0=[200, 950, 6])
print(popt)

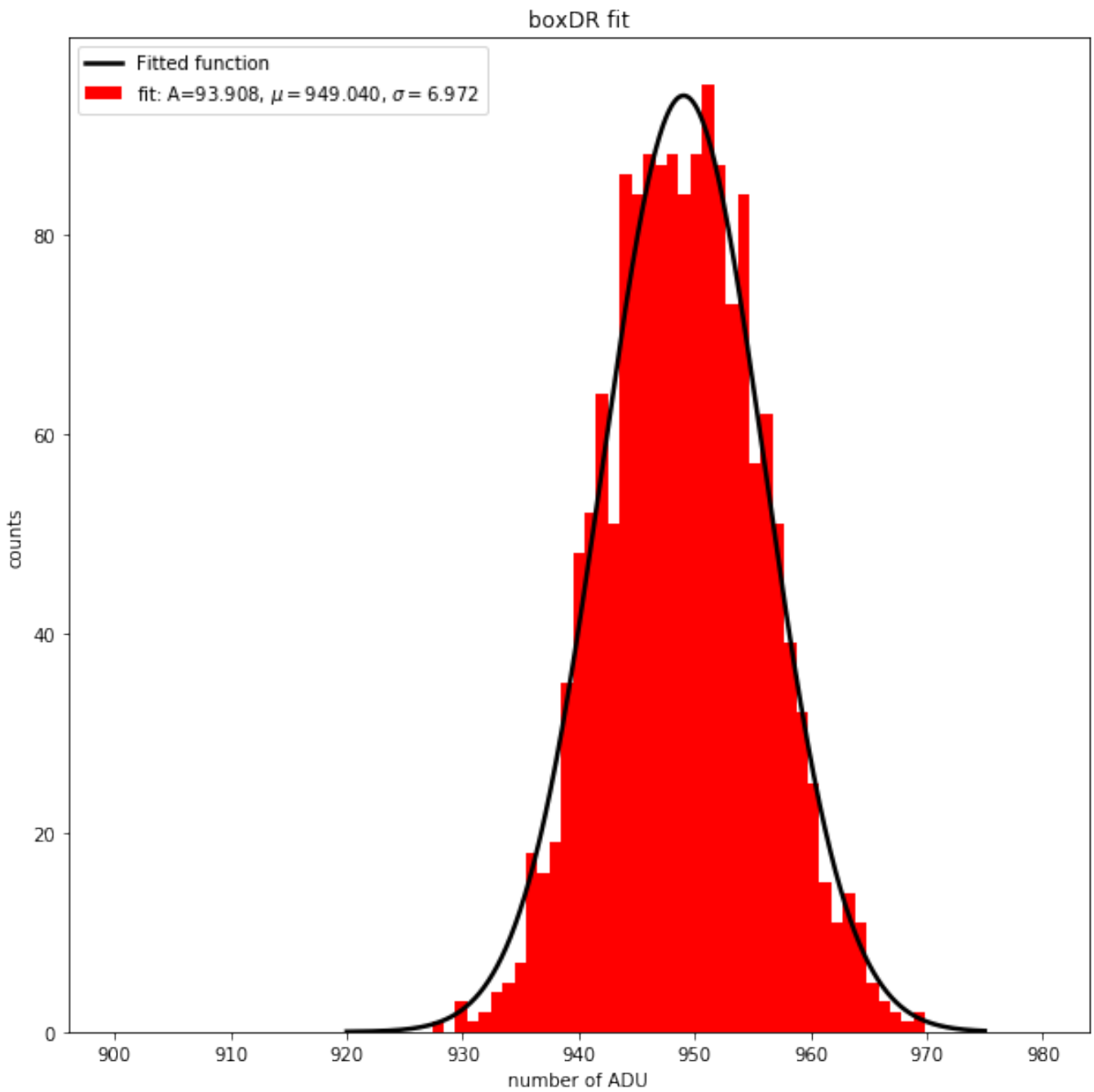
# Generate enough x values to make the curves look smooth.
xspace = np.linspace(920, 975, 10000)

# Plot the histogram and the fitted function.
plt.bar(binscenters, data_entries, width=bins[1] - bins[0], color='red', label='fit: A=%5.3f',
plt.plot(xspace, fit_function(xspace, *popt), color='k', linewidth=2.5, label=r'Fitted function')
plt.title("boxDR fit")
plt.xlabel("number of ADU")
plt.ylabel("counts")
plt.legend(loc='best')
plt.show()

poptDR = tuple(popt)

```

```
[ 93.90814027  949.040279    6.97239603]
```



In [222]: *#find the average sigma between the 4 boxes*

```
list_of_sigmas = [poptUL[2], poptDL[2], poptUR[2], poptDR[2]]
average_of_sigmas = np.mean(list_of_sigmas)
print('sigma average of background:', average_of_sigmas)

list_of_mus = [poptUL[1], poptDL[1], poptUR[1], poptDR[1]]
```

```
average_of_mus = np.mean(list_of_mus)
print('mu average of background:', average_of_mus)
```

sigma average of background: 6.572267235291228

mu average of background: 942.7927938062426