```
pip install astropy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Requirement already satisfied: astropy in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/di
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: pyerfa>=1.7.3 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packag
```

```
import numpy as np
import math
from scipy import integrate
import matplotlib.pyplot as plt
import matplotlib.colors as colors
import matplotlib.ticker as plticker
import random
import cv2
from astropy.io import fits
```

```
img_c8 = fits.open("d0221_0099.fits.gz")[8].data
```

```
# Opening all the frames
img_100 = fits.open("d0221_0100.fits.gz")[8].data
img_101 = fits.open("d0221_0101.fits.gz")[8].data
img_102 = fits.open("d0221_0102.fits.gz")[8].data
img_103 = fits.open("d0221_0103.fits.gz")[8].data
```

PART TWO OF LAB CONTINUED

Examine the data from chip 8 in the other 4 dark frames (frames 100-103). For each image, using the same standard deviation computed above (as measured on the subset of the image with counts within plus or minus 50 of the median pixel value), what fraction of pixels are 10 standard deviations above below the median pixel value? Across all 5 frames, are there any pixels that are always above or below 10 standard deviations of the median pixel value?

```
small_std = 2.5835420823057285
```

```
# Finding the median value of each frame
m100 = np.median(img_100)
m101 = np.median(img_101)
m102 = np.median(img_102)
m103 = np.median(img_103)
```

```
# Frame 100
```

```
af100 = np.array(np.where(img_100 > (10*small_std)+m100))
bf100 = np.array(np.where(img_100 < m100-(10*small_std)))

a100size = np.size(af100)
b100size = np.size(bf100)

n_100 = 100*((a100size + b100size)/8739612)
# n_100 gives fraction of pixels that lie outside of 10 sigma,
# meaning that they are either above or below 10 sigma

print(n_100)

# 100 - n_100 = 99.39493881421738
# The above expression calculates the fraction of pixels that lie within
# (above or below) 10 standard deviations of the median pixel value
```

```
0.6050611857826182
```

```
# Frame 101
```

```
af101 = np.array(np.where(img_101 > (10*small_std)+m101))
bf101 = np.array(np.where(img_101 < m101-(10*small_std)))

a101size = np.size(af101)
b101size = np.size(bf101)

n_101 = 100*((a101size + b101size)/8739612)
# Gives fraction of pixels that lie outside of 10 sigma,
# meaning that they are either above or below 10 sigma

print(n_101)
```

```
0.5976009003603363
```

```
# Frame 102
```

```
af102 = np.array(np.where(img_102 > (10*small_std)+m102))
bf102 = np.array(np.where(img_102 < m102-(10*small_std)))

a102size = np.size(af102)
b102size = np.size(bf102)

n_102 = 100*((a102size + b102size)/8739612)
# Gives fraction of pixels that lie outside of 10 sigma,
```

```
# meaning that they are either above or below 10 sigma

print(n_102)
```

```
0.5982416610714526
```

```
# Frame 103
```

```
af103 = np.array(np.where(img_103 > (10*small_std)+m103))
bf103 = np.array(np.where(img_103 < m103-(10*small_std)))

a103size = np.size(af103)
b103size = np.size(bf103)

n_103 = 100*((a103size + b103size)/8739612)
# Gives fraction of pixels that lie outside of 10 sigma,
# meaning that they are either above or below 10 sigma

print(n_103)
```

```
0.6111712968493338
```

```
# Create a table that shows the fraction of pixels outside of 10stds
# for each of the specified frames.
```

```
np.array(af103)[0].shape
```

```
(26707,)
```

```
#pixel_xy_103 = []

#for k in range(0,26707):
#  pixel_xy_103.append((af103[1],af103[0]])
```

```
#holder = np.zeros((4096,2140,5))
```

```
#holder[0:4096,0:2140,0] =  fits.open("d0221_0099.fits.gz")[8].data[0:4096,0:2140]
#holder[0:4096,0:2140,1] =  fits.open("d0221_0100.fits.gz")[8].data[0:4096,0:2140]
#holder[0:4096,0:2140,2]  =  fits.open("d0221_0101.fits.gz")[8].data[0:4096,0:2140]
#holder[0:4096,0:2140,3]  =  fits.open("d0221_0102.fits.gz")[8].data[0:4096,0:2140]
#holder[0:4096,0:2140,4]  =  fits.open("d0221_0103.fits.gz")[8].data[0:4096,0:2140]
```

```
#for k in range(0,5):
#  plt.imshow(holder[:,:,k])
# title = 'Frame' + str(k)
 # insert np.where line of code to find the pixels in each frame that are above
```

```
  # and below 10 sigma
#  plt.title(title)
#  plt.show()
```

Finding the number of pixels in each frame that are above/below 10 sigma.

```
# Frame 99
```

```
sig99 = np.array(np.where(img_c8 > np.median(img_c8) + 10*small_std))
np.array(sig99).shape[1]/(img_c8.shape[0]*img_c8.shape[1])*100
```

```
0.3069440895151869
```

```
#Frame 100
```

```
sig100 = np.array(np.where(img_100 > m100 + 10*small_std))
np.array(sig100).shape[1]/(img_100.shape[0]*img_100.shape[1])*100
```

```
0.30163916471962615
```

```
# Frame 101
```

```
sig101 = np.array(np.where(img_101 > m101 + 10*small_std))
np.array(sig101).shape[1]/(img_101.shape[0]*img_101.shape[1])*100
```

```
0.29792001314252337
```

```
sig101b = np.array(np.where(img_101 < 10*small_std - m101))
np.array(sig101b).shape[1]/(img_101.shape[0]*img_101.shape[1])*100
```

```
0.0
```

```
# Frame 102
```

```
sig102 = np.array(np.where(img_102 > m102 + 10*small_std))
np.array(sig102).shape[1]/(img_102.shape[0]*img_102.shape[1])*100
```

```
0.2982394494742991
```

```
sig102b = np.array(np.where(img_101 < 10*small_std - m102))
np.array(sig102b).shape[1]/(img_101.shape[0]*img_101.shape[1])*100
```

```
0.0
```

```
np.min(img_101)
```

```
983
```

```
m102 - 10*small_std
```

```
969.1645791769428
```

```
# Frame 103
```

```
sig103 = np.array(np.where(img_103 > m103 + 10*small_std))
np.array(sig103).shape[1]/(img_103.shape[0]*img_103.shape[1])*100
```

```
0.30468521831191586
```

```
sig100.shape
```

```
(2, 26440)
```

```
# The following block of code is an attempt to use a for loop to find all the
# pixels that are outside of 10sigma in every frame.
```

```
intersections = []

for i in sig99:
  if i in sig100 and sig101 and sig102 and sig103:
    intersections.append(i)

print(intersections)
```

```
[]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DeprecationWarni
  after removing the cwd from sys.path.
```

```
# The following block of code is an attempt to use the np.intersect1d() function
# to find all the pixels that are outside of 10sigma in every dark frame.
```

```
sig99.flatten
sig100.flatten
sig101.flatten
sig102.flatten
sig103.flatten
```

```
<function ndarray.flatten>
```

```
poop = np.intersect1d(sig99,sig100)
```

```
poop.shape
```

```
(4096,)
```

```
# The following block of code is yet another attempt at using a for loop to
# find all the pixels that are outside of 10sigma in every frame.
```

```
q = [x for x in sig99 if x in sig100 and x in sig101 and x in sig102 and x in sig10
print(q)
```

```
[]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarni
  """Entry point for launching an IPython kernel.
```

```
sig99.shape
```

```
(2, 26905)
```

```
sig100.shape
```

```
(2, 26440)
```

```
sig101.shape, sig102.shape, sig103.shape
```

```
((2, 26114), (2, 26142), (2, 26707))
```

```
# Reshape arrays to be of similar size
```

```
sig99t = np.transpose(sig99)
sig100t = np.transpose(sig100)
sig101t = np.transpose(sig101)
sig102t = np.transpose(sig102)
sig103t = np.transpose(sig103)
```

```
sig99t.shape
```

```
(26905, 2)
```

```
# The following block of code is how I actually found all the pixels that are
# outside 10sigma. A detailed explanation of how this code works is included
# in my written lab report.
```

```
q = np.array([x for x in sig99t if x in sig100t])
o = np.array([x for x in q if x in sig101t])
```

```
w = np.array([x for x in o if x in sig102t])
v = np.array([x for x in w if x in sig103t])

print(v)
# v is an array with all the common pixels outside of 10sigma
```

```
[[   0 1126]
 [   0 1127]
 [   0 1128]
 ...
 [4095 1722]
 [4095 1723]
 [4095 1724]]
```

```
v.shape
```

```
(26905, 2)
```