07/09/2019

Gaussian fit

get the photon count for each pixel in a column of an image

make a plot that is “Photon Count vs. Row Count” for the column that you’ve calculated

the plot should be a Gaussian fit

repeat for all columns in an image, save those plots together

repeat plots for all images and consolidate accordingly

\*\*\*might be useful to make histogram first, then fit to gaussian

\*\*\*or do both a gaussian and a histogram for each column in each image

======

# import packages to use

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.pyplot import \*

from astropy.utils.data import get\_pkg\_data\_filename

from astropy.table import Table

from astropy.io import fits

from scipy.optimize import curve\_fit

from scipy import asarray as ar,exp

import glob

import batman

import lmfit

import corner

import matplotlib.pyplot as plt

from matplotlib import pylab

from numpy import arange,array,ones

from scipy import stats

import numpy.polynomial.polynomial as poly

directory='/home/ian/Desktop/WebbData/visit23\_defringed/'

#directory='/Users/annaburkholder/exp\_det\_scripts/visit23\_defringed/'

number\_of\_rows=64

number\_of\_columns=1024

number\_of\_images=43

#Load images into a list

list=glob.glob(directory+"\*.fits")

#print, first image in list

print(list[0])

##Example load first fits image

hdul=fits.open(list[1])

#Get MJD mid time of exposure from Header, which has start and end MJD times

mjd\_start=hdul[0].header['EXPSTART']

mjd\_end=hdul[0].header['EXPEND']

mjd=(mjd\_end+mjd\_start)/2.

print(mjd)

#load fits file image into an array called 'data'

data = hdul[0].data

data.shape #size of image

data.dtype.name #type of image

print(np.sum(data)) #total counts in image

#close fits after loading in data needed

hdul.close()

index\_of\_images=np.arange(number\_of\_images)

index\_of\_rows=np.arange(number\_of\_rows)

index\_of\_columns=np.arange(number\_of\_columns)

total\_counts=np.zeros((number\_of\_images))

mjd=np.zeros((number\_of\_images))

row\_sum=np.zeros(number\_of\_rows) #sums pixel count for each row of image

maxRow=np.zeros(number\_of\_columns) #number of row in which max count exists in cols

col\_vals=np.zeros(number\_of\_columns)

col\_max=np.zeros(number\_of\_columns) #to contain max photon count per column in an image

col\_posn\_max=np.zeros(number\_of\_columns) #to contain position of max photon count per column in an image

col\_max\_posn=np.zeros((number\_of\_images, number\_of\_columns)) #contains all column pc among all images

col\_max\_value=np.zeros((number\_of\_images, number\_of\_columns)) #contains all position pc among all images

**for** i **in** index\_of\_images:

img=list[i]

hdul=fits.open(img)

data = hdul[0].data

**for** j **in** index\_of\_columns:

col\_test= data[0:64,j]

*#print(col\_test)*

*#plot each count value for the column (not Gaussian fit yet)*

x = np.arange(0,64)

y = col\_test

plt.plot (x,y)

*#plt.show()*

*#set up plotting params for Gaussian fit*

mean = sum(x \* y) / sum(y)

sigma = np.sqrt(sum(y \* (x - mean)\*\*2) / sum(y))

**def** Gauss(x, a, x0, sigma):

**return** a \* np.exp(-(x - x0)\*\*2 / (2 \* sigma\*\*2))

popt,pcov = curve\_fit(Gauss, x, y, p0=[max(y), mean, sigma])

plt.plot(x, y)

plt.plot(x, Gauss(x, \*popt),'r-')

*#plt.show()*

plt.savefig('/Users/annaburkholder/exp\_det\_scripts/plots/Col'+str(j)+'Im'+str(i)+'.png')

plt.close()

print(np.argmax(Gauss(x, \*popt)))

print(np.max(Gauss(x, \*popt)))

======

*#script that fits a Gaussian to each column of each image for an exoplanet*

*# import packages to use*

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**from** matplotlib.pyplot **import** \*

**from** astropy.utils.data **import** get\_pkg\_data\_filename

**from** astropy.table **import** Table

**from** astropy.io **import** fits

**from** scipy.optimize **import** curve\_fit

**from** scipy **import** asarray **as** ar,exp

**import** glob

**import** batman

**import** lmfit

**import** corner

**import** matplotlib.pyplot **as** plt

**from** matplotlib **import** pylab

**from** numpy **import** arange,array,ones

**from** scipy **import** stats

**import** numpy.polynomial.polynomial **as** poly

*#directory='/home/ian/Desktop/WebbData/visit23\_defringed/'*

directory='/Users/annaburkholder/exp\_det\_scripts/visit23\_defringed/'

number\_of\_rows=64

number\_of\_columns=1024

number\_of\_images=43

*#Load images into a list*

list=glob.glob(directory+"\*.fits")

*#print, first image in list*

**print**(list[0])

*##Example load first fits image*

hdul=fits.open(list[1])

*#Get MJD mid time of exposure from Header, which has start and end MJD times*

mjd\_start=hdul[0].header['EXPSTART']

mjd\_end=hdul[0].header['EXPEND']

mjd=(mjd\_end+mjd\_start)/2.

**print**(mjd)

*#load fits file image into an array called 'data'*

data = hdul[0].data

data.shape *#size of image*

data.dtype.name *#type of image*

**print**(np.sum(data)) *#total counts in image*

*#close fits after loading in data needed*

hdul.close()

index\_of\_images=np.arange(number\_of\_images)

index\_of\_rows=np.arange(number\_of\_rows)

index\_of\_columns=np.arange(number\_of\_columns)

total\_counts=np.zeros((number\_of\_images))

mjd=np.zeros((number\_of\_images))

row\_sum=np.zeros(number\_of\_rows) *#sums pixel count for each row of image*

maxRow=np.zeros(number\_of\_columns) *#number of row in which max count exists in cols*

col\_vals=np.zeros(number\_of\_columns)

col\_max=np.zeros(number\_of\_columns) *#to contain max photon count per column in an image*

col\_posn\_max=np.zeros(number\_of\_columns) *#to contain position of max photon count per column in an image*

col\_max\_posn=np.zeros((number\_of\_images, number\_of\_columns)) *#contains all column pc among all images*

col\_max\_value=np.zeros((number\_of\_images, number\_of\_columns)) *#contains all position pc among all images*

*#loop through every column in every image in order to obtain all gaussian fits*

**for** i **in** index\_of\_images:

img=list[i]

hdul=fits.open(img)

data = hdul[0].data

**for** j **in** index\_of\_columns:

col\_test= data[0:64,j]

*#print(col\_test)*

*#plot each count value for the column (not Gaussian fit yet)*

x = np.arange(0,64)

y = col\_test

plt.plot (x,y)

*#plt.show()*

*#set up plotting params for Gaussian fit*

mean = sum(x \* y) / sum(y)

sigma = np.sqrt(sum(y \* (x - mean)\*\*2) / sum(y))

**def** Gauss(x, a, x0, sigma):

**return** a \* np.exp(-(x - x0)\*\*2 / (2 \* sigma\*\*2))

popt,pcov = curve\_fit(Gauss, x, y, p0=[max(y), mean, sigma])

plt.plot(x, y)

plt.plot(x, Gauss(x, \*popt),'r-')

*#plt.show()*

plt.savefig('/Users/annaburkholder/exp\_det\_scripts/plots/Col'+str(j)+'Im'+str(i)+'.png') *#saves respective col/im plots to 'plots' folder*

plt.close()

*#print(np.argmax(Gauss(x, \*popt))) #returns index of max count in column*

*#print(np.max(Gauss(x, \*popt))) #returns value of max count in column*

=============

=============

=============

# import packages to use

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.pyplot import \*

from astropy.utils.data import get\_pkg\_data\_filename

from astropy.table import Table

from astropy.io import fits

from scipy.optimize import curve\_fit

from scipy import asarray as ar,exp

import glob

import batman

import lmfit

import corner

import matplotlib.pyplot as plt

from matplotlib import pylab

from numpy import arange,array,ones

from scipy import stats

import numpy.polynomial.polynomial as poly

from scipy.optimize import curve\_fit, minimize\_scalar

directory='/home/ian/Desktop/WebbData/visit23\_defringed/'

#directory='/Users/annaburkholder/exp\_det\_scripts/visit23\_defringed/'

number\_of\_rows=64

number\_of\_columns=1024

number\_of\_images=43

#Load images into a list

list=glob.glob(directory+"\*.fits")

#print, first image in list

print(list[0])

##Example load first fits image

hdul=fits.open(list[1])

#Get MJD mid time of exposure from Header, which has start and end MJD times

mjd\_start=hdul[0].header['EXPSTART']

mjd\_end=hdul[0].header['EXPEND']

mjd=(mjd\_end+mjd\_start)/2.

print(mjd)

#load fits file image into an array called 'data'

data = hdul[0].data

data.shape #size of image

data.dtype.name #type of image

print(np.sum(data)) #total counts in image

#close fits after loading in data needed

hdul.close()

index\_of\_images=np.arange(number\_of\_images)

index\_of\_rows=np.arange(number\_of\_rows)

index\_of\_columns=np.arange(number\_of\_columns)

total\_counts=np.zeros((number\_of\_images))

mjd=np.zeros((number\_of\_images))

row\_sum=np.zeros(number\_of\_rows) #sums pixel count for each row of image

maxRow=np.zeros(number\_of\_columns) #number of row in which max count exists in cols

col\_vals=np.zeros(number\_of\_columns)

col\_max=np.zeros(number\_of\_columns) #to contain max photon count per column in an image

col\_posn\_max=np.zeros(number\_of\_columns) #to contain position of max photon count per column in an image

col\_max\_posn=np.zeros((number\_of\_images, number\_of\_columns)) #contains all column pc among all images

col\_max\_value=np.zeros((number\_of\_images, number\_of\_columns)) #contains all position pc among all images

gauss\_max\_value=np.zeros((number\_of\_images, number\_of\_columns))

gauss\_max\_posn=np.zeros((number\_of\_images, number\_of\_columns))

col\_gauss = np.zeros(number\_of\_rows) #Create arrays for gaussian fits

col\_test= data[0:64,30]

x = np.arange(0,64)

y = col\_gauss

def Gauss(x, a, x0, sigma): return a \* np.exp(-(x - x0)\*\*2 / (2 \* sigma\*\*2))

for i in index\_of\_images:

img=list[i]

#print(img)

hdul=fits.open(img)

mjd\_start=hdul[0].header['EXPSTART']

mjd\_end=hdul[0].header['EXPEND']

mjd\_image=(mjd\_end+mjd\_start)/2.

mjd[i]=mjd\_image

data = hdul[0].data

#print(np.sum(data)) #total counts in image

for j in index\_of\_columns:

#total\_counts[i]=np.sum(data[0:64,j]) #total counts in column j of image

col\_gauss = data[0:64,j]

mean = sum(x \* col\_gauss) / sum(col\_gauss)

sigma = np.sqrt(sum(col\_gauss \* (x - mean)\*\*2) / sum(col\_gauss))

popt,pcov = curve\_fit(Gauss, x, col\_gauss, p0=[max(y), mean, sigma])

col\_posn\_max[j]=np.argmax(Gauss(x, \*popt))

col\_max[j]=np.max(Gauss(x, \*popt))

#fm = lambda x: -Gauss(x, \*popt)

#r = minimize\_scalar(fm, bounds=(1, 128))

#print("maximum:", Gauss(r["x"], \*popt))

gmv=popt[0]

gmp=popt[1]

gauss\_max\_value[i][j]=gmv

gauss\_max\_posn[i][j]=gmp

#print(total\_counts)

#print(row\_vals) #prints array for each image w/ max photon count per column

#col\_max\_posn[i]=total\_counts

#col\_max\_value[i]=col\_max

#print(col\_max\_value)

#col\_max\_posn[i]=col\_posn\_max

#print(col\_max\_posn)

print(gauss\_max\_value[i])

print(gauss\_max\_posn[i])