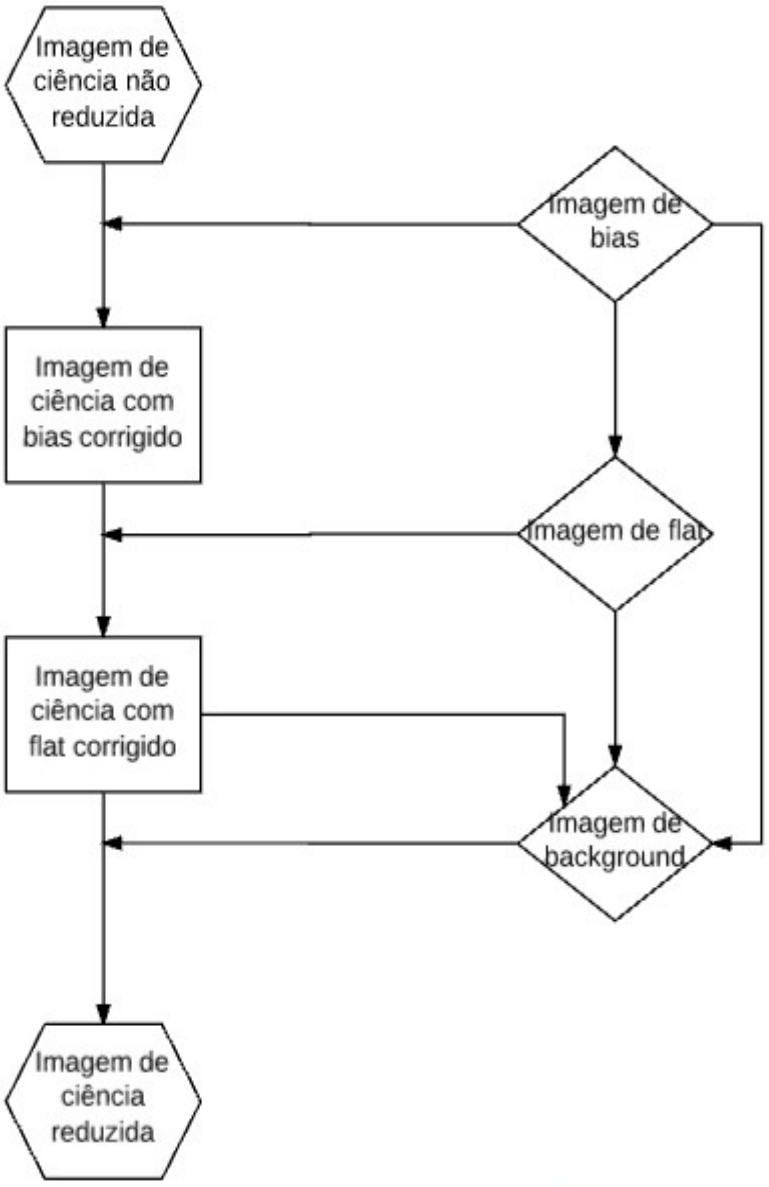
Tratamento de Dados Astronômicos – Projeto 1
Ariane Serrano Zarro
2017.2
Prazo de entrega: 15/09

1) Introdução

O primeiro projeto da disciplina de tratamento de dados astronômicos tem como objetivo criar um pipeline (programa com sequencia de passos pelos quais dados observados passam para serem reduzidos) para corrigir imagens de ciência através das imagens de bias, flatfield e background. A linguagem utilizada para construir o programa será python (versão 2.7) com auxílio dos pacotes adequados. O fluxograma abaixo ilustra o processo pelo qual as imagens, ainda cruas, passarão até estarem completamente corrigidas.



Os pacotes necessários para realizar o trabalho serão:

```
>>import numpy as np #manipulação de matrizes.
>>import matplotlib.pyplot as plt #plot das imagens.
>>from astropy.io import fits #leitura dos arquivos fits.
>>import glob #listar arquivos de diretórios.
```

Obs.: O arquivo.py com os códigos precisa estar no mesmo diretório das imagens de ciência, bias e flat.

2) Desenvolvimento do Projeto

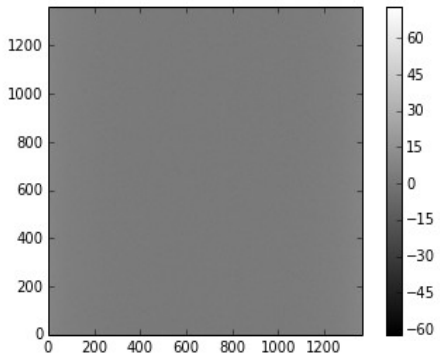
2.1) Imagem de Bias

Bias é um termo usado para descrever uma variação de pixel-a-pixel no ponto zero de uma câmera CCD. Cada pixel tem uma pequena diferença na base de valores e esta diferença leva o nome de bias, que é removido subtraindo a imagem de bias da imagem de ciência. A imagem de bias basicamente subtrai o viés do CCD. O bias também precisa ser retirado da imagem de flat e de background, como indica o fluxograma acima, pois o viés do equipamento também os afetará.

A função para calcular o bias final, com número indeterminado de imagens de bias, é o bias() e está apresentado e explicado abaixo:

```
>>def bias(): #função de cálculo do bias
     arraymbias =np.array([])  #array vazio para ser usado fora do laço for.
     master_bias = [] #variável vazia que ainda será utilizada.
     for i in glob.glob('/home/ariane/fitsfile/bias.B.*.fits'): #glob-leitura dos arquivos padronizados. Nesse caso, só serão lidos os arquivos com o nome 'bias.B.*.fits' sendo * o número da imagem. Fitsfile é o diretório em que estão os arquivos fits.
          b_img, hdr = fits.getdata(i, header=True) #lendo as imagens bias e seus headers.
          b_img = np.array(b_img, dtype='Float64') #só para inserir o 'float64' mesmo.
          master_bias.append(b_img) #juntar as imagens bias do filtro B.
    master_bias = np.array(master_bias) #transformar em array.
    arraymbias2 = np.vstack([arraymbias, master_bias]) if arraymbias.size else master_bias # colocar o array 'master_bias' dentro do array vazio 'arraymbias' que está fora do laço for. Fizemos isso com a função 'vstack' do numpy.
    arraymbias2 = np.array(arraymbias2, dtype='Float64') #só para inserir o 'float64' mesmo.
    b_mediana = np.median(arraymbias2, axis=0) #calcular a mediana da imagem de bias.
    b_mediana = np.array(b_mediana, dtype='Float64') #só para inserir o 'float64' mesmo.
    plt.figure() #plotar o bias
    plt.imshow(b_mediana,vmin=np.mean(b_mediana)-2.5*np.std(b_mediana), vmax=np.mean(b_mediana) + 2.5*np.std(b_mediana), cmap=plt.cm.gray, origin='lower')
    plt.colorbar()
    plt.show()
>>return(b_mediana) #retornar a matriz final do bias.
```

A função bias() retornará a imagem final do bias e a matriz correspondente. O resultado está exemplificado abaixo:




```
Out[44]: array([[ 287.09567261,  288.43109131,  288.09628296, ...,  317.15054321,
                   314.9147644 ,  314.9928894 ],
                 [ 287.79431152,  286.50576782,  286.49328613, ...,  313.07952881,
                   315.17562866,  315.28067017],
                 [ 286.9175415 ,  286.90869141,  287.59484863, ...,  316.10891724,
                   314.44839478,  314.47311401],
                 ...,
                 [ 285.47119141,  286.2303772 ,  286.85089111, ...,  312.75241089,
                   315.2739563 ,  315.29788208],
                 [ 285.73690796,  286.86508179,  285.75009155, ...,  313.840271  ,
                   314.45941162,  314.47415161],
                 [ 285.69833374,  285.74697876,  285.37484741, ...,  313.93972778,
                   313.55096436,  313.63745117]])
```

2.2) Imagem de Flat

O objetivo do flatfield é corrigir os problemas causados pela variação na sensibilidade pixel a pixel do detector e/ou por distorções no caminho óptico. O bias também será subtraído da imagem de flatfield como mencionado anteriormente.

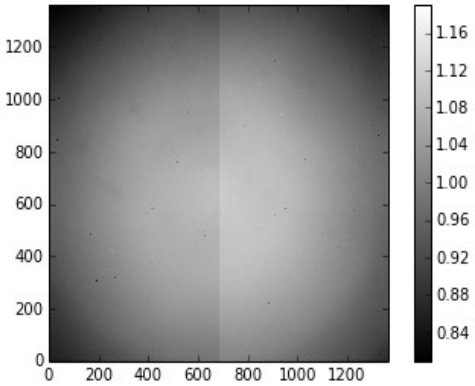
A função para calcular o flat final, com um número indeterminado de imagens de flat, é flat_bias(bias) que terá a função bias, construída anteriormente, como argumento. O cálculo do flat é um pouco diferente do cálculo de bias como está apresentado e explicado abaixo:

```
>>def flat_bias(bias): #função de cálculo do flat final com o bias retirado. Por isso o input será bias.
        bias1 = np.array(bias(), dtype = 'Float64')
        arraymflat =np.array([])  #array vazio para ser usado fora do laço for.
        master_flat = [] #variável vazia que ainda será utilizada.
        for i in glob.glob('/home/ariane/fitsfile/flat.B.*.fits'): #glob-leitura dos arquivos padronizados.
Nesse caso, só serão lidos os arquivos com o nome 'flat.B.*.fits' sendo * o número da imagem.
fitsfile é o diretório em que estão os arquivos fits.
            f_img, hdr = fits.getdata(i, header=True) #lendo as imagens flat e seus headers.
            f_img = np.array(f_img, dtype='Float64') #só para inserir o 'float64' mesmo.
#primeiro devemos subtrair o bias do flat para depois fazer o master flat e a normalização.
            flat_bias = f_img - bias1 #subtraindo o bias final de cada imagem de flat.
            flat_bias = np.array(flat_bias, dtype='Float64')
#agora podemos normalizar,ou seja, dividir o flat(com bias subtraído) pela média do flat(com bias
subtraído) e depois juntar as imagens flat num master flat.
            flat_bias_media = np.mean(flat_bias)
            flat_norm = flat_bias/flat_bias_media
            master_flat.append(flat_norm)
        master_flat = np.array(master_flat) #transformar em array.
       arraymflat2 = np.vstack([arraymflat, master_flat]) if arraymflat.size else master_flat # colocar
o   array 'master_flat' dentro do array vazio 'arraymflat' que está fora do laço for. Fizemos isso com
a função 'vstack' do numpy.
```

```
        arraymflat2 = np.array(arraymflat2, dtype='Float64')
        f_mediana = np.median(arraymflat2, axis=0)
        plt.figure() #plotar o bias
        plt.imshow(f_mediana,vmin=np.mean(f_mediana)-2.5*np.std(f_mediana),
vmax=np.mean(f_mediana)+2.5*np.std(f_mediana), cmap=plt.cm.gray, origin='lower')
        plt.colorbar()
        plt.show()
   >> return(f_mediana) #retornar a matriz final do bias.
```

A função flat_bias() retornará a imagem do master bias, a imagem do flat final e o array correspondente a imagem do flat final corrigido pelo bias. O resultado está exemplificado abaixo:

Imagem do flat final e o array do flat final:




```
Out[190]: array([[ 1.09279282,  0.31396411,  0.31967505, ...,  0.33771913,
                   1.26309926,  1.26349917],
                 [ 0.91083355,  0.22461917,  0.22878924, ...,  0.22547107,
                   1.11600917,  1.11635242],
                 [ 0.92580824,  0.25487242,  0.25564955, ...,  0.25848462,
                   1.20281123,  1.2031652 ],
                 ...,
                 [ 0.98344795,  0.26192626,  0.26262143, ...,  0.26890419,
                   1.1826589 ,  1.182935  ],
                 [ 0.96007433,  0.26243451,  0.26279355, ...,  0.27348994,
                   1.16273197,  1.16299888],
                 [ 0.91005165,  0.26442122,  0.26288552, ...,  0.27018231,
                   1.07904405,  1.07929528]])
```

2.3) Imagem de Background (Sky Image)

A imagem de background (ou sky image) será retirada da imagem de ciência com a imagem de bias e flat já retiradas. O cálculo do background é bastante distinto dos outros anteriores pois é retirada de uma região de uma imagem de ciência já corrigida pelo bias e pelo flat. Dessa forma, será necessário que o usuário insira algumas informações durante a execução do programa entre elas a área da imagem de ciência da qual será extraída a imagem do céu.

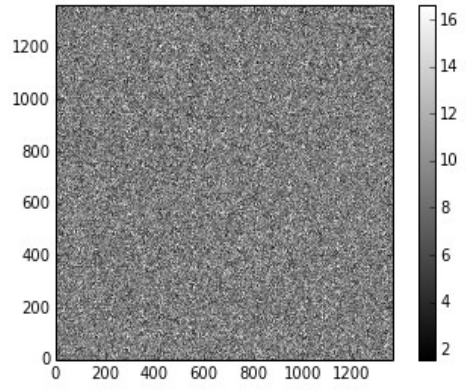
A função que realiza o cálculo do background é o sky() com as funções bias() e flat_bias() como argumento. O código está exposto abaixo:

```
>>def sky(bias, flat_bias): #cáculo da imagem do céu a partir da imagem corrigida pelo bias e o flat.
        bias1 = np.array(bias(), dtype = 'Float64')
        flat_bias1 = np.array(flat_bias(bias), dtype = 'Float64')
        sky_img, hdr = fits.getdata(input('Imagem de ciência para retirar o sky image: '), header = input('Header da imagem: True ou False? ')) #usuário escolherá a imagem e indicará se o header está presente ou não.
        sky_img = np.array(sky_img ,dtype = 'Float64')
        sky_bias = sky_img - bias1 #tirar o bias da imagem de ciência.
        sky_bias_flat = sky_bias/flat_bias1

#agora temos que perguntar ao usuário que valores de x e y ele deseja 'recortar' da imagem de ciência para fazer a imagem do céu.
        size = sky_img.shape #dimensão do array.
        print('Escolha os valores de x e y entre 0 e %s: '%str(size[1]))
        xi = int(input('xi= '))
        xf = int(input('xf= '))
        yi = int(input('yi= '))
        yf = int(input('yf= '))
        sky_bias_flat = sky_bias_flat[yi:yf, xi:xf]

#depois de dados os valores, calcular a função de poisson correspondente.
        sky_image = np.random.poisson(np.mean(sky_bias_flat), size) #size: int or tuple of ints.
        plt.figure() #plotar a imagem do céu.
        plt.imshow(sky_image,vmin=np.mean(sky_image)-2.5*np.std(sky_image), vmax=np.mean(sky_image)+2.5*np.std(sky_image), cmap=plt.cm.gray,origin='lower')
        plt.colorbar()
        plt.show()
    >>return(sky_image) #retornar a matriz final da imagem do céu.
```

O resultado da imagem do céu e o array correspondente encontra-se abaixo:




```
Out[37]: array([[ 7, 15,  9, ..., 11, 15,  5],
                [10, 10,  8, ...,  8, 16, 11],
                [15,  5,  8, ...,  6,  8, 16],
                ...,
                [ 8, 10,  8, ..., 14, 12, 17],
                [10,  9,  8, ...,  7,  8,  8],
                [ 6,  7, 16, ...,  4,  4,  5]])
```

2.4) Correção das imagens de ciência

Finalmente, temos o cálculo da imagem de ciência completamente corrigida. A função final tem as 3 funções construídas anteriormente como argumento e retornará a imagem de bias, a imagem de flat, a imagem do céu e as imagens de ciência corrigidas. As imagens de ciências serão salvas em arquivos fits no mesmo diretório das imagens cruas.

```
>>def img_final(bias, flat_bias, sky): #cálculo das imagens de ciência corrigidas.
        bias1 = np.array(bias(), dtype = 'Float64')
        flat_bias1 = np.array(flat_bias(bias), dtype = 'Float64')
        sky_bias_flat1 = np.array(sky(bias,flat_bias), dtype = 'Float64')
        for i in glob.glob('/home/ariane/fitsfile/xo2b.*.fits'):
            img_fin, hdr = fits.getdata(i, header=True)
            img_fin = np.array(img_fin, dtype='Float64')
            hdr = fits.getheader(i) #salvar imagem.
            outfile = '_bfs_corrigida.fits'

            b_img_fin = img_fin - bias1 #corrigir o bias da imagem de ciência.
            f_b_img_fin = b_img_fin/flat_bias1 #corrigir o flat da imagem de ciência.
            s_f_b_img_fin = f_b_img_fin - sky_bias_flat1 #corrigir o background da imagem de
ciência.

            hdu = fits.PrimaryHDU() #criando o header.
            hdu.data = s_f_b_img_fin
            hdu.header = hdr #header do arquivo corrigido igual ao antigo com algumas flags
adicionadas abaixo.
```

Obs.: Não pude terminar o código porque, por algum motivo, a minha versão do python não consegue executar a última parte do pipeline.

3) Referências

https://tellescopio.comh.br/calibracao-de-imagens-light-dark-flat-bias-frames
https://en.wikipedia.org/wiki/Flat-field_correction
https://pythonhelp.wordpress.com/2012/08/20/glob-listando-arquivos-de-diretorios/
https://waltersmartinsf.github.io/TDAclass/codes/OVL474--TrabalhandoFITS.html
https://waltersmartinsf.github.io/TDAclass/docs/OVL474_02_Pipelines.pdf
https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.shape.html
https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.poisson.html
https://stackoverflow.com/questions/2050637/appending-the-same-string-to-a-list-of-strings-in-python