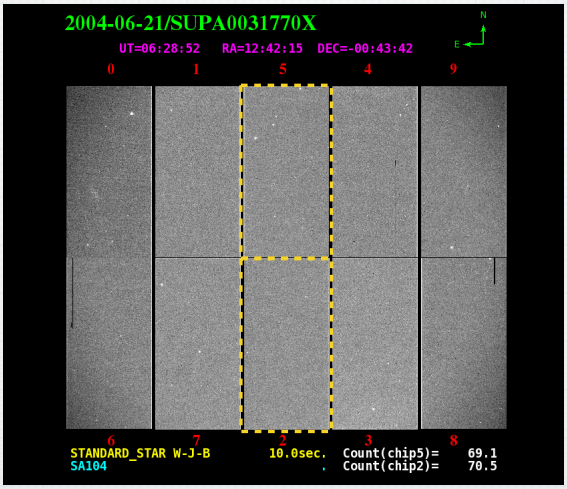
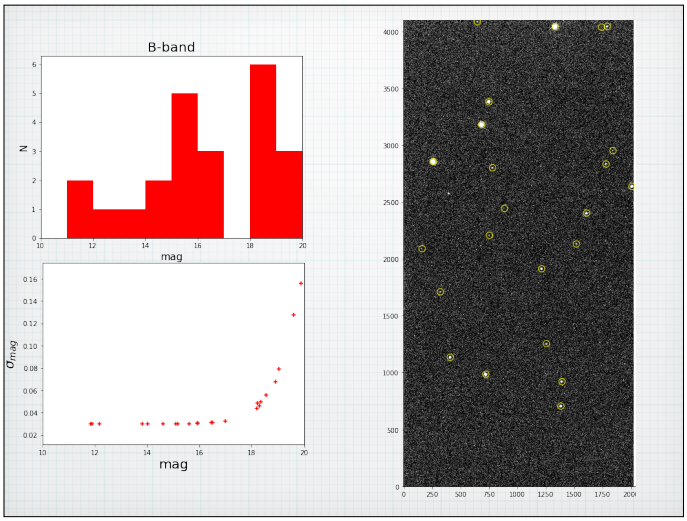
at 国立天文台

Python+Jupyter notebookによる光赤外撮像データ処理入門

2018-08-30, 31　中島 康





Jupyter notebookの使い方

- Jupyter notebookの起動
- 新規ノートブックファイルの作成
- Pythonとの対話
- 対話を保存
- Jupyter notebookの終了

メモの書き込み

- Markdownで書き込み

講習0 : Pythonとnotebook

- notebook形式の資料の使い方
- Pythonについて少し
- notebookでの変数の読み込み
- ipythonの補完機能

ipython の補完機能　~ [tab]キー

最初の何文字か　+ [tab] ⇨ 候補のリスト
- 定義済み変数
- 使用可能な関数

など

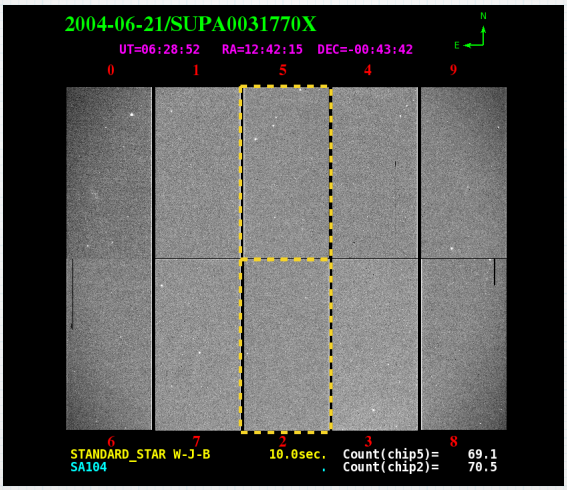
変数.何文字か　+ [tab] ⇨ 候補のリスト
- 使用可能なメソッド

ドット

0文字なら全候補

講習1 : IRAFタスクをPythonから使ってみる

- IRAFとPythonをつなぐPyRAF
- IRAFの基本タスク(コマンド)
  display, imexam, imstat, imarith
- IRAFタスクのパラメータ設定
- IRAFタスクから戻り値を取得
- 演習1



FITSファイル	OBJECT	フィルター	積分時間(秒)
data1/SUPA003175[0-6-5].fits	ドームフラット	B	10
data1/SUPA00317705.fits	target1	B	10
data1/SUPA00317885.fits	target2	B	30
data2/SUPA003175[0-6-2].fits	ドームフラット	B	10
data2/SUPA00317702.fits	target1	B	10
data2/SUPA00317882.fits	target2	B	30

from pyraf import iraf

あるいは　import pyraf.iraf as iraf

DS9にFITSを表示

ds9をあらかじめ起動しておく！

対象ファイル

iraf.display('xxxxxx.fits', 1)

ds9での表示Frame番号
(なしの場合、直前で使った番号)

imexamで画像を調べる

対象ファイル　　DS9で表示されている Frame番号

iraf.imexam('xxxxxx.fits', 1)

m : 5x5ピクセルのカウント値の統計
a : 星の特徴量(ピーク値、FWHMなど)
r : 星のradial profile
q : 終了

imstatで画像の統計を調べる

対象ファイル

iraf.imstat('xxxxxx.fits')

irafタスクのパラメータ設定

iraf.epar('imstat') 1. GUIで設定

iraf.imstat('xxxxxx.fits', fields='mean, stddev')

2. 関数の引数として書き込む

iraf.imstat.fields = 'midpt, mean, stddev'
iraf.imstat.lower = 0

3. 変数に代入

パラメータのリセット

iraf.imstat.unlearn( )

または

iraf.unlearn('imstat')

戻り値を変数へ

out = **iraf.imstat**('xxxxxx.fits', Stdout=1)

標準出力へ出る内容が変数に(文字列として)

format='no'　ヘッダ行を非表示

画像の四則演算

数値 または ファイル
対象ファイル

iraf.imarith('xxxxxx.fits', '-', 100, 'out.fits')

演算子　　出力ファイル

IRAFが勝手にFITS拡張するのを禁止する

iraf.reset(fkinit = 'append = no')

helpドキュメントを読む

iraf.help('imstat')

演習1

data1/SUPA00317885.fitsはtarget2の生データです。
新しいノートブックファイルを作成し、

1. imexamでバックグラウンドの値とばらつき、星の特徴量、を調べる。
2. imstatでカウント値のmedian, mean, standard deviationを求める。
3. imarithで、全pixelからmedian値を引く。(元のファイルに上書きせずに、別名のfitsファイルに)

講習2 : IRAFで1次処理

生データ取得 ＋ 一次処理に必要なデータ取得

天体以外の信号の除去
感度と光学系のムラの補正
→ 一次処理

一次処理済みデータ

例) 点光源の測光 (＋ 標準星データ取得)

測光値の較正

可視光

raw(x, y) = flat(x, y) x object(x, y) + dark(x, y) + bias(x, y)

- flat(x,y) : ピクセル間の**感度むら**, 光学系の**透過むら**
  - object(x,y)=const. の光がやってきたとしても flat(x,y)×const. が検出される。
- 近年のCCDではdarkは無視できるレベル
- biasは、raw(x, y)のオーバースキャン領域から推測 (x方向への依存は無しと仮定し)

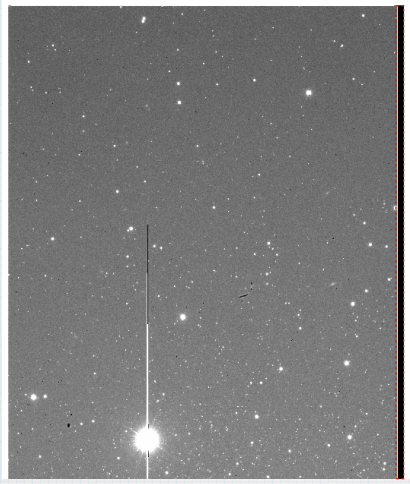
raw(x, y) = flat(x, y) x object(x, y) + dark(x, y) + bias(x, y)

flat(x, y)				object(x, y)						
1.1	0.9	0.9	×	200	204	201	=	220	183	181
1.0	1.2	0.8		198	400	202		198	480	162
1.1	0.7	1.1		205	199	200		226	139	220

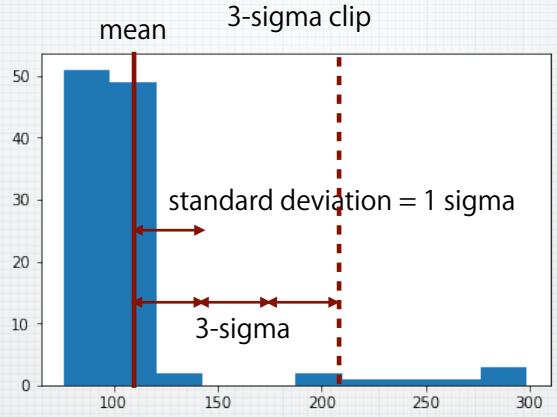
可視光

raw(x, y) = flat(x, y) x object(x, y) + dark(x, y) + bias(x, y)

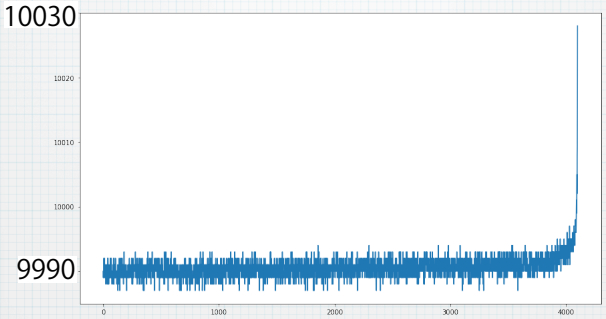
- flat(x,y) : ピクセル間の**感度むら**, 光学系の**透過むら**
  - object(x,y)=const. の光がやってきたとしても flat(x,y)×const. が検出される。
- 近年のCCDではdarkは無視できるレベル
- biasは、raw(x, y)のオーバースキャン領域から推測 (x方向への依存は無しと仮定し)



ここがオーバースキャン領域



Y方向にも大部分で一様で、端のほうで **0.数%** 増加



フラット

raw(x, y) = flat(x, y) x object(x, y) + dark(x, y) + bias(x, y)

object(x, y) = constant (一様光)を観測してやれば、

$$\text{flat}(x, y) = \frac{\text{raw}(x, y) - \text{bias}(x, y)}{\text{constant}}$$

constant = median(raw(x, y) - bias(x, y))
規格化

オーバースキャンのメジアン

iraf.imstat('data1/SUPA00317505.fits[2049:2080, *]')

光が当たっている部分のメジアン

iraf.imstat('data1/SUPA00317505.fits[1:2048, *]')

1次処理　ドームフラットデータから flat(x, y)

天体生データから bias(x, y)

$$\text{object}(x, y) = \frac{\text{raw}(x, y) - \text{bias}(x, y)}{\text{flat}(x, y)}$$

iraf.imcombine()

im01.fits

101	104	102
99	100	101
103	97	100

im02.fits

105	101	99
100	99	102
101	98	96

im03.fits

98	102	105
103	102	99
105	101	103

averageでcombine

101.3	102.3	102
100.7	100.3	100.7
103	98.7	99.7

medianでcombine

101	102	102
100	100	101
103	98	100

iraf.imcombine()

im01.fits

101	104	102
99	100	101
103	97	100

im02.fits

105	101	99
100	99	102
101	98	96

im03.fits

98	102	105
103	102	99
105	101	103

```
iraf.imcombine(input='im01.fits, im02.fits, im03.fits',
               output='test.fits', combine='median')
```

combine='average'

演習2

2-1

target2を観測した、'./data1/SUPA00317885.fits' について、バイアス引き+フラット割りの処理をしましょう。これは5番フレームです。フィルターも同じBバンドなので、フラット割りには、'bflatn5.fits'が使えます。

この結果のフレームを'btarget2n5.fits'と呼ぶことにします。(後の演習で利用します)

2-2

2番フレームの生データ'./data2/SUPA00317702.fits' について、

バイアス引き+フラット割りの処理をしましょう。

先ほどの5番フレームとは違い、これは2番フレームなので、2番フレームのためのフラットを作成する必要があります。

(1) './data2/SUPA00317502.fits' を規格化したものをフラットとして作成する。

(2) './data2/SUPA003175[0-6]2.fits' から平均のフラットを作成する。

(3) 上のどちらかのフラットを使って、バイアス引き後のフラット割りを行う。

注意 : 2番フレームはオーバースキャン領域が5番とは異なる。

at 国立天文台

2日目

Python+Jupyter notebookによる光赤外撮像データ処理入門

2018-08-31　中島 康

講習3 : 星の測光

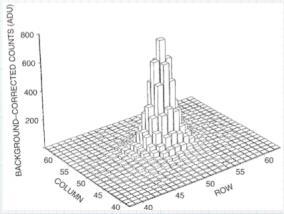
- 視野の星を検出させ、 iraf.daofind()
- それらをアパーチャ測光　iraf.phot()
- 測光値の較正 ~ 標準星
- 演習3

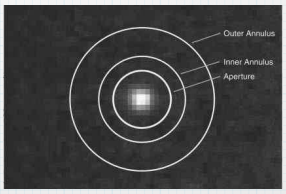
星の測光

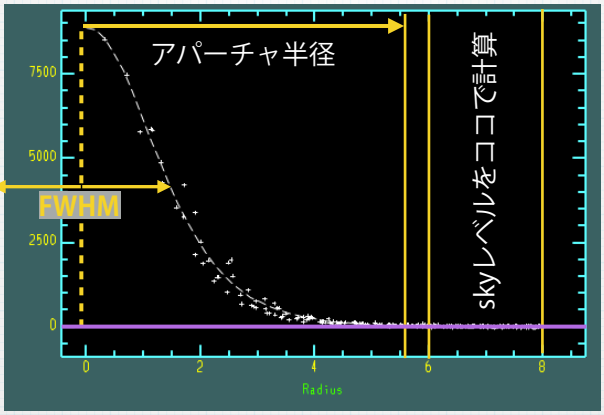
アパーチャ測光

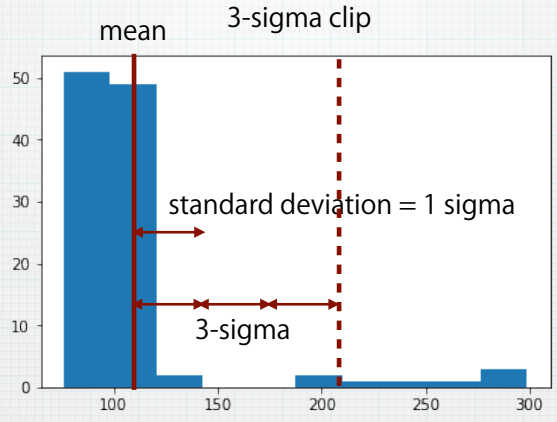
- 星を含む円内のカウントを総計する。
- 明るさの分かっている星のカウントと比べて、明るさを求める。

標準星の観測が必要









iraf.phot()のパラメータ設定

```
med = 69.  # 背景レベルとばらつき、fwhm
std = 7.4
fwhm = 7.0

iraf.apphot.unlearn()  # デフォルト値に戻しておく

iraf.apphot.datapars.datamax = 50000  # サチった星を数えない
iraf.apphot.datapars.readnoise = 10 # 検出器に特有な値
iraf.apphot.datapars.epadu = 2.5 # 検出器に特有な値
iraf.apphot.datapars.itime = 10  # 積分時間

iraf.apphot.findpars.threshold = 7  # 7シグマ以上のものを検出せよ
iraf.apphot.findpars.sharphi = 0.8  # 星っぽくないものを除くため

# fwhmで決まるパラメータ
iraf.apphot.datapars.fwhmpsf = fwhm
iraf.apphot.centerpars.cbox = max(5.0, fwhm)
iraf.apphot.fitskypars.annulus = 3 * fwhm
iraf.apphot.photpars.apertures = 2 * fwhm

iraf.apphot.fitskypars.dannulus = 10.

# 背景のレベルとばらつきで決まるパラメータ
iraf.apphot.datapars.sigma = std
iraf.apphot.datapars.datamin = med - 5 * std

iraf.apphot.photpars.zmag = 27  # 等級のゼロ点
```

測光対象ファイル

iraf.daofind('xxxxxx.fits', output='out1.coo')

検出結果のテキストファイル

daofindの結果
のファイル

測光対象ファイル

iraf.phot('xxxx.fits', coords='out1.coo'
output='out1.mag')

測光結果を書き出す
(テキスト)ファイル

iraf.photの
測光結果ファイル

iraf.txdump('out1.mag',
fields='xc, yc, mag, merr', Stdout='result1.txt')

抽出する項目

この結果の
書き出し先

演習3

演習2-1で処理をしたbtarget2n5.fitsで測光をしてみましょう。このときも、オーバースキャン領域などの不要な部分を削除して行いましょう。

btarget2n5.fits'btarget2n5.fits'の視野の中には測光標準星は写っていません。ただし、上のtarget1と近い時間に観測したデータですので、等級ゼロ点は同じだと仮定し、上と同じ較正値(1.672)を使ってください。

講習4 : Numpyの基本

- 配列のベクトル演算 ~ ndarray
- 二次元配列の注意点
- numpyの関数をいくつか

ndarrayの作成

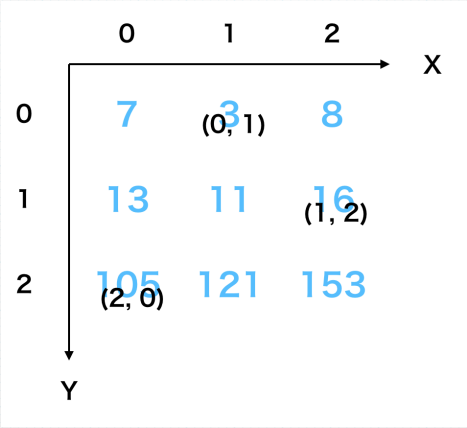
np.array(［リスト])

ndarrayの演算(ベクトル演算)

```
  np.array([1, 2, 3])
+ np.array([10, 20, 30])
= np.array([11, 22, 33])
```

2次元のndarray は インデックスが(y, x)

```
np.array([  [7, 3, 8], [13,11,16], [105, 121, 153]  ])
np.array([  [7, 3, 8],
            [13,11,16],
  [105, 121, 153]  ])
```



矩形部分の抽出

iraf.imstat('data1/SUPA00317505.fits[2049:2080, *]')

array[:, 2048:2080]　または　**array**[:, 2048:]

1. 区間ワイルドカードは ：
2. IRAFのインデックスは1はじまり
   Pythonのインデックスは0はじまり
3. numpyでは (y, x)

離散部分の抽出

```
aa = np.array([0, 10, 20, 30, 40, 50])
aa[ np.array([0, 2, 5]) ]
```
→ array([0, 20, 50])

```
aa = np.array([0, 10, 20, 30, 40, 50])
aa[  aa > 30 ]
```
→ array([40, 50])

```
aa = np.array([0, 10, 20, 30, 40, 50])
aa[  aa > 30 ]
```
→ array([40, 50])

```
aa = np.array([0, 10, 20, 30, 40, 50])
aa[  aa > 30 ]  = -99
aa
```
→ array[(0, 10, 20, 30, -99, -99)]

numpy.where( )　　numpyの関数をいくつか

```
aa = np.array([0, 10, 20, 30, 40, 50])
x = np.where(aa > 30)
x
```
→ (array([4, 5]),)

aa[x] → array([40, 50])

numpy.stack( )

```
aa = np.array([1, 2, 3])
bb = np.array([4, 5, 6])
cc = np.array([10, 20, 30])
s1 = np.stack((aa, bb, cc))
s1 → array([[1,2,3],
            [4, 5, 6],
            [10, 20, 30]])
```

stack!

numpy.median([スタックされたndarray], axis=0)

```
s1 → array([[1,2,3],
            [4, 5, 6],
            [10, 20, 30]])
```

axis=0

np.median(s1, axis=0) → array([4., 5., 6.])

np.average(s1, axis=0) → array([5., 9., 13.,])

np.sum(s1, axis=0) → array([15., 27., 39.,])

numpy.append( )

```
aa = np.array([1, 2, 3])
bb = np.array([4, 5, 6])
cc = np.array([10, 20, 30])
s2 = aa[np.newaxis, : ]
s2 = np.append(s2, bb[np.newaxis, :], axis=0)
s2 = np.append(s2, cc[np.newaxis, :], axis=0)
s2 → array([[1,2,3],
            [4, 5, 6],
            [10, 20, 30]])
```

numpy.empty( )

```
aa = np.array([1, 2, 3])
bb = np.array([4, 5, 6])
cc = np.array([10, 20, 30])
s3 = np.empty((0, 3))        0 x 3の2次元ndarray
s3 = np.append(s3, aa[np.newaxis, :], axis=0)
s3 = np.append(s3, bb[np.newaxis, :], axis=0)
s3 = np.append(s3, cc[np.newaxis, :], axis=0)
s3 → array([[1,2,3],
            [4, 5, 6],
            [10, 20, 30]])
```

numpy.loadtxt( )

数値(半角)だけのテーブル(サイズ)であれば、これを使うのが楽。(ヘッダ行に文字があっても、無視できる。) forループを回さなくて良い！

numpy.savetxt( )

演習4

(1) 下の二次元配列の演算をnumpyを用いて行ってください。

10	11	12
20	21	22
30	31	32

＋

100	110	120
200	210	220
300	310	320

(2) 下の3つの二次元配列のメジアンを、numpy.stack()とnumpy.median()を用いて求めてください。

10	11	12
20	21	22
30	31	32

100	110	120
200	210	220
300	310	320

20	22	24
40	42	44
60	62	64

講習5 : astropy.io.fitsの基本

- IRAFなしでFITSの処理
- getdata()でFITSの読み込み
- numpyを使って処理
- writeto()でFITSの書き出し

モジュールのよびだし

from astropy.io import fits

(画像)データの読み込み

imdata = fits.getdata('xxxxx.fits')

ヘッダの読み込み

header = fits.getheader('xxxxx.fits')

データの書き出し(新規)

fits.writeto('newname.fits', imdata, header)

データの書き出し(更新)

fits.update('exsiting.fits', imdata, header)

フラットの作成

pyraf版

```
from pyraf import iraf

iraf.imstat.unlearn()
iraf.imstat.fields='midpt'

img = './data1/SUPA00317505.fits'
out1 = iraf.imstat(img + '[2049:2080, *]', format='no', Stdout=1)
out2 = iraf.imstat(img + '[1:2048, *]', format='no', Stdout=1)
med1 = float(out1[0])  # bias
med2 = float(out2[0])

iraf.imarith(img, '-' , med1, 'bflat505.fits')  # バイアス値を引く
iraf.imarith('bflat505.fits', '/' , med2-med1, 'bflat505.fits')  # 規格化
```

numpy版

```
import numpy as np
from astropy.io import fits

imdata = fits.getdata('./data1/SUPA00317505.fits')
omed = np.median(imdata[:, 2049:])  # オーバースキャン部分のメジアン
imed = np.median(imdata[:, :2048])  # 光が当たる部分のメジアン
imdata = ( imdata - omed ) / (imed - omed)  # ゲタを引き、規格化
fits.writeto('bflat505ap.fits', imdata)
```

演習5

1. 上のmy10x10.fitsを作成し、DS9で見てxとyが反転していることを確かめてください。

2. astropy.io.fitsとnumpyを使って以下の処理をしてください。

(1) './data2/SUPA003175[0-6]2.fits'から2番フレーム用のフラットを作成。

(2) './data2/SUPA00317882.fits' について、バイアス引き+フラット割りの処理

(3) trimmingして(2)の結果からオーバスキャン部をとりのぞく。

講習6 : matplotlibの基本

- ヒストグラム (光度関数)
- X-Y プロット (等級 vs 等級エラー)
- FITS画像の表示
- FITS画像にオーバープロット

モジュールのよびだし

```
import matplotlib.pyplot  as plt
```

notebook内に描画するために

```
%matplotlib inline
```

ヒストグラム

```
plt.hist( ndarray )
put.show()
```

基本

データプロット

```
plt.scatter( ndarray, ndarray )
put.show()
```

基本

FITSの表示

```
plt.figure( figsize=(7, 14) )
plt.imshow(imdata, plt.cm.gray,
           origin='lower', interpolation='none')
put.show()
```

基本

演習6

演習3で行った'btarget2n5.fits'の測光結果を用いて、

(1)「光度関数のヒストグラム」と「等級vs等級エラーのプロット」を作成してください。

(2) FITS画像をnotebookに表示して、そこに測光した星をプロットしてください。