PDMXray

November 27, 2020

[68]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from astropy.io import fits
from bokeh.plotting import output_notebook, figure, show
from bokeh.models import HoverTool, tools,ColumnDataSource, Whisker, ColorBar,␣
 ↪LinearColorMapper
from astropy.modeling import models, fitting
from PyAstronomy.pyTiming import pyPDM
from bokeh.palettes import viridis, inferno

output_notebook()
```

[69]:
```
dataxray = fits.open('U18_lc.flc')
datatable = dataxray[1].data
datatable
```

[69]:
```
FITS_rec([(0.000000e+00, 50., 0., 0., 1.),
          (1.000000e+02, 50., 0., 0., 1.),
          (2.000000e+02, 50., 0., 0., 1.), …,
          (2.196605e+08, 50., 0., 0., 1.),
          (2.196606e+08, 50., 0., 0., 1.),
          (2.196607e+08, 50., 0., 0., 1.)],
         dtype=(numpy.record, [('TIME', '>f8'), ('XAX_E', '>f8'), ('RATE1',
'>f4'), ('ERROR1', '>f4'), ('FRACEXP', '>f4')]))
```

[70]:
```
dataxray[1].header
```

[70]:
```
XTENSION= 'BINTABLE'           / binary table extension
BITPIX  =                    8 / 8-bit bytes
NAXIS   =                    2 / 2-dimensional binary table
NAXIS1  =                   28 / width of table in bytes
NAXIS2  =              2196608 / number of rows in table
PCOUNT  =                    0 / size of special data area
GCOUNT  =                    1 / one data group (required keyword)
TFIELDS =                    5 / number of fields in each row
TTYPE1  = 'TIME    '           / label for field   1
```

```
TFORM1  = 'D       '           / data format of field: 8-byte DOUBLE
TUNIT1  = 's       '           / physical unit of field
TTYPE2  = 'XAX_E   '           / label for field   2
TFORM2  = 'D       '           / data format of field: 8-byte DOUBLE
TTYPE3  = 'RATE1   '           / label for field   3
TFORM3  = 'E       '           / data format of field: 4-byte REAL
TUNIT3  = 'count/s '           / physical unit of field
TTYPE4  = 'ERROR1  '           / label for field   4
TFORM4  = 'E       '           / data format of field: 4-byte REAL
TUNIT4  = 'count/s '           / physical unit of field
TTYPE5  = 'FRACEXP '           / label for field   5
TFORM5  = 'E       '           / data format of field: 4-byte REAL
EXTNAME = 'RATE    '           / name of this binary table extension
DATE    = '2020-08-07T21:01:22' / file creation date (YYYY-MM-DDThh:mm:ss UT)
CREATOR = 'lcurve 1.0 (xronos5.22)' / Name of XRONOS program that created this f
HDUCLASS= 'OGIP    '
HDUCLAS1= 'LIGHT CURVE'
HDUCLAS2= 'TOTAL   '
HDUCLAS3= 'RATE    '
CONTENT = 'XRONOS OUTPUT'
ORIGIN  = 'HEASARC/GSFC'
OBJECT  = 'NGC6397 '
TIMVERSN= 'OGIP/93-003'
TSTARTI =                11756 / Start time for this extension
TSTARTF =   0.6469172485176387 / Start time for this extension
TSTOPI  =                14299 / Stop time for this extension
TSTOPF  =   0.0172865829918010 / Stop time for this extension
TIMEUNIT= 'd       '           / Units for header timing keywords
TIMEZERI=                11756 / Zero-point offset for TIME column
TIMEZERF=   0.6474959522220161 / Zero-point offset for TIME column
COMMENT TIMESYS keyword is not currently set. If the input lightcurve
COMMENT contains the header keyword MJDREF, the time in the xronos
COMMENT  output is in TJD (JD-2440000.5)
TIMEDEL = 1.1574074074074073E-03
AVRGE_1 =       7.85908569E-03 / Avg, count/s in interval
FREXP_1 =       1.60747848E-03 / Avg. Frac exposure in frame
VAROB_1 =       8.68914503E-05 / observed variance
VAREX_1 =       7.85900120E-05 / expected variance
THRDM_1 =       9.97775146E-07 / third moment
MININ_1 =       0.00000000E+00 / Minimun Intensity
MAXIN_1 =       5.99999987E-02 / Maximun Intensity
EXVAR_1 =       8.30143836E-06 / excess of variance
CHI2_1  =       3.90395483E+03 / Chi squared
RMS_1   =       3.66610289E-01 /  RMS variability
AVRGE_E1=       1.49209445E-04 / Error Avg count/s
VAROB_E1=       2.06825780E-06 / Error observed variance
VAREX_E1=       1.87066064E-06 / Error expected variance
```

```
CHI2_E1 =       8.40238037E+01 / Error Chi2
RMS_E1  =       4.56694737E-02 / RMS variability
```

[71]:
```python
name = 'u18Xray'
mjd = datatable['TIME']
mag = datatable['RATE1']
dmag = datatable['ERROR1']
```

[9]:
```python
mag.min()
```

[9]:
```
nan
```

[54]:
```python
mjd[0:4]
```

[54]:
```
array([  0., 100., 200., 300.])
```

[33]:
```python
limits =␣
↪[(0,500),(562500,563000),(564299,564649),(2174909,2175800),(2195089,2196599)]
```

[34]:
```python
newmjd =[]
newmag = []
newdmag = []
for l in limits:
    low,high = l
    newmjd.append(mjd[low:high])
    newmag.append(mag[low:high])
    newdmag.append(dmag[low:high])
newmjd = np.array(newmjd)
newmag = np.array(newmag)
newdmag = np.array(newdmag)
```

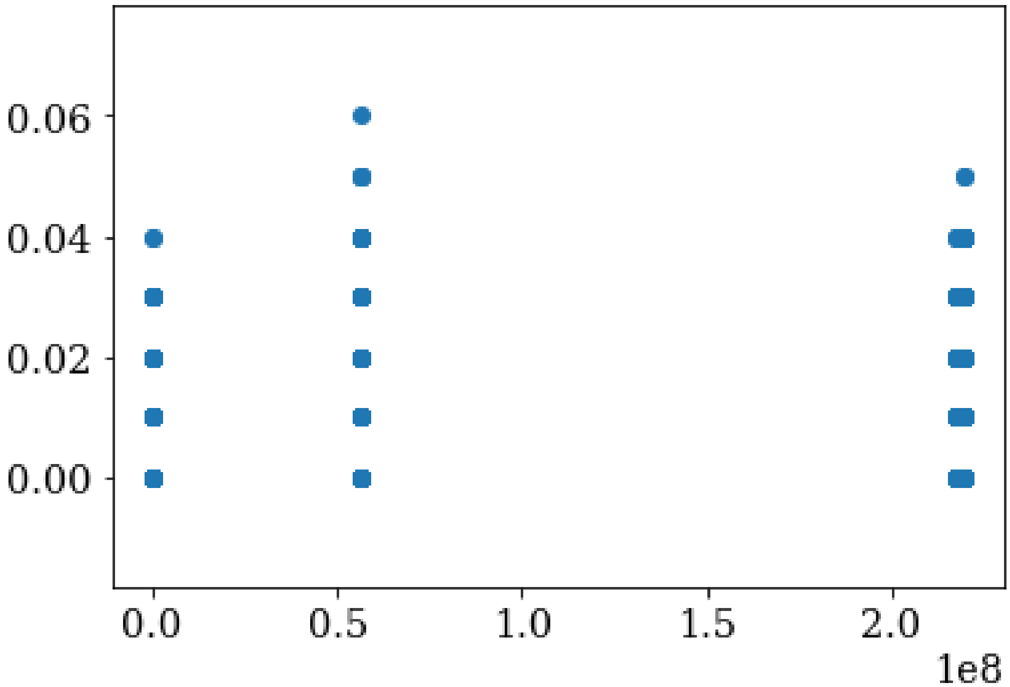
[35]:
```python
flatmjd = []
for sublist in newmjd:
    for item in sublist:
        flatmjd.append(item)
```

[36]:
```python
flatmag = []
for sublist in newmag:
    for item in sublist:
        flatmag.append(item)
```

[37]:
```python
flatmjd = np.array(flatmjd)
flatmag = np.array(flatmag)
flatmjd = flatmjd[~np.isnan(flatmag)]
flatmag = flatmag[~np.isnan(flatmag)]
```

[38]:
```python
plt.scatter(flatmjd,flatmag)
```

[38]: <matplotlib.collections.PathCollection at 0x7f138ca304d0>



[15]:
```
print(len(flatmjd))
```

3483

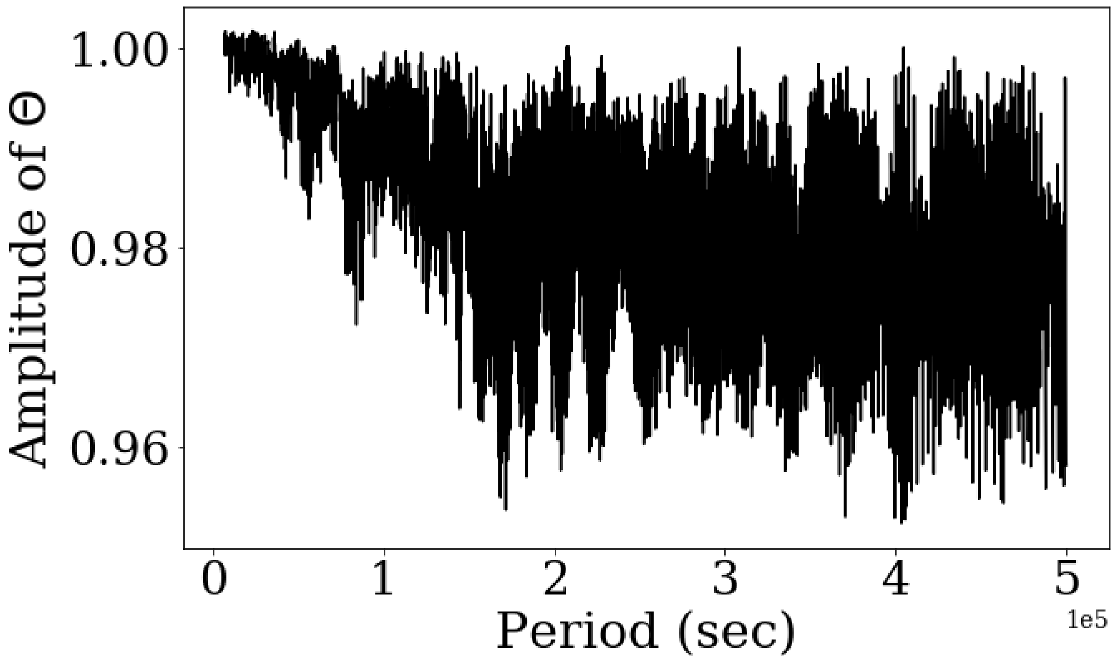
[16]:
```
print(len(flatmjd))
```

3483

[17]:
```
# Get a ``scanner'', which defines the frequency interval to be checked.
# Alternatively, also periods could be used instead of frequency.
S = pyPDM.Scanner(minVal=7200., maxVal=500000. ,dVal=100., mode="period")

# Carry out PDM analysis. Get frequency array
# (f, note that it is frequency, because the scanner's
# mode is ``frequency'') and associated Theta statistic (t).
# Use 10 phase bins and 3 covers (= phase-shifted set of bins).
P = pyPDM.PyPDM(flatmjd, flatmag)
#f1, t1 = P.pdmEquiBin(10,S)

f1, t1 = P.pdmEquiBinCover(10, 3, S)
```

```
# Show the result
```

[25]:
```
thetadic = {'theta':t1,
            'period':f1}



# plot the periodogram

plt.rc('font', family='serif')
plt.rc('xtick', labelsize='x-large')
plt.rc('ytick', labelsize='x-large')


fig = plt.figure(figsize=(10, 6))

ax = fig.add_subplot(1, 1, 1)
#ax.set(xlim=(0.2, 10),
#          ylim=(0, 1));
ax.set_xlabel('Period (sec)',fontsize=30)
ax.set_ylabel('Amplitude of $\Theta$',fontsize=30)
ax.tick_params(axis='both', which='major', labelsize=28)






plt.plot(thetadic['period'],thetadic['theta'],color='k',ls='solid')

plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
plt.show()
#save iamge

#save iamge
#fig.savefig('PDMXRAY.eps', format='eps',bbox_inches = "tight")
fig.savefig('PDMXRAYAllLC.png', format='png',bbox_inches = "tight")
```



[19]:
```
# Show the result

p = figure(plot_width=500, plot_height=500, title='',active_drag='pan',␣
 ↪active_scroll='wheel_zoom',
           y_axis_label='Theta',x_axis_label='Period')

thetadic = {'theta':t1,
           'period':f1}


        #Tool to get wavelength
hover2 = HoverTool(
        tooltips=[
            ('Date', '(@period)')
        ]
    )
p.add_tools(hover2)
p.line(x='period',y='theta',source=thetadic)
show(p)
```

1 Search a LC onlyn

[115]:
```
limits = np.
 ↪array([(0,500),(562500,563000),(564299,564649),(2174909,2175800),(2195089,2196599)])
newmjd =[]
newmag = []
newdmag = []
select =[0,1,2,3,4]
for l in limits[select]:
    low,high = l
    newmjd.append(mjd[low:high])
    newmag.append(mag[low:high])
    newdmag.append(dmag[low:high])
newmjd = np.array(newmjd)
newmag = np.array(newmag)
newdmag = np.array(newdmag)

flatmjd = []
for sublist in newmjd:
    for item in sublist:
        flatmjd.append(item)

flatmag = []
for sublist in newmag:
    for item in sublist:
        flatmag.append(item)

flatdmag = []
for sublist in newdmag:
    for item in sublist:
        flatdmag.append(item)
flatmjd = np.array(flatmjd)
flatmag = np.array(flatdmag)
flatdmag = np.array(flatdmag)

flatmjd = flatmjd[~np.isnan(flatmag)]
flatmag = flatmag[~np.isnan(flatmag)]
flatdmag = flatdmag[~np.isnan(flatdmag)]

print(len(flatmag))
```

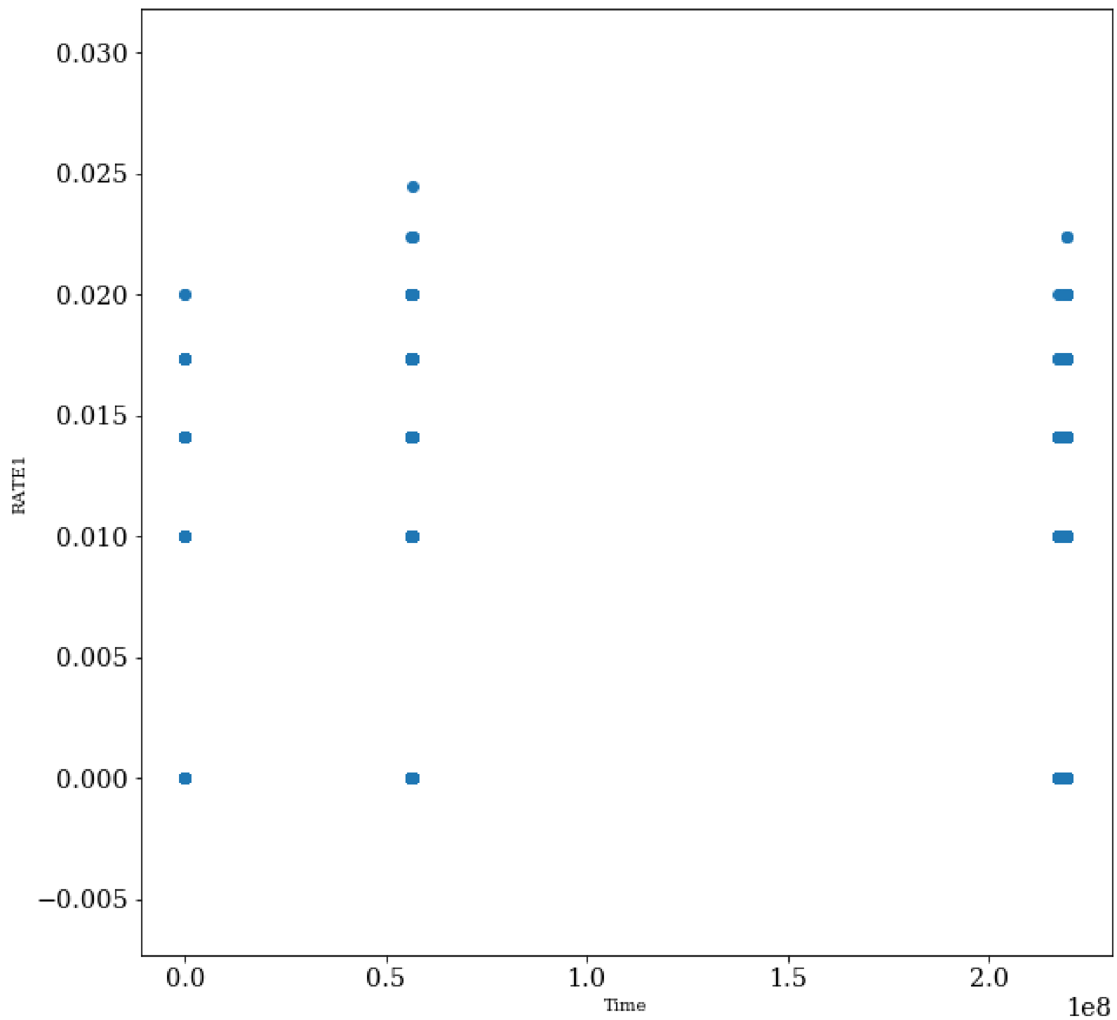
3483

[116]:
```
with open('../PDM/TuCas.dat','w') as f:
    f.write('Time Val Sig\n')
    for i,j,k in zip(flatmjd,flatmag,flatdmag):
        f.write(f'{i} {j} {k}\n')
```

[117]:

```
plt.figure(figsize=(10,10))
plt.scatter(flatmjd,flatmag)
plt.xlabel('Time')
plt.ylabel('RATE1')
```

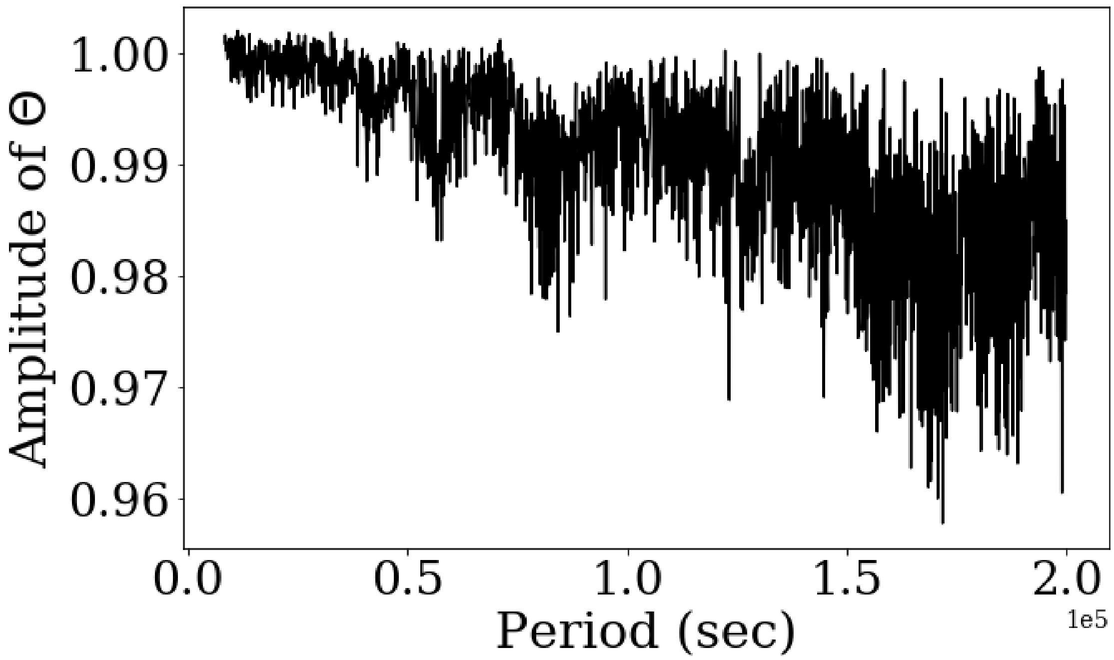
[117]: Text(0, 0.5, 'RATE1')



[118]:

```
# Get a ``scanner'', which defines the frequency interval to be checked.
# Alternatively, also periods could be used instead of frequency.
S = pyPDM.Scanner(minVal=8400., maxVal=200000. ,dVal=100., mode="period")

# Carry out PDM analysis. Get frequency array
# (f, note that it is frequency, because the scanner's
```

```
# mode is ``frequency'') and associated Theta statistic (t).
# Use 10 phase bins and 3 covers (= phase-shifted set of bins).
P = pyPDM.PyPDM(flatmjd, flatmag)
f1, t1 = P.pdmEquiBin(10,S)

#f1, t1 = P.pdmEquiBinCover(10, 3, S)


# Show the result

thetadic = {'theta':t1,
           'period':f1}


# plot the periodogram

plt.rc('font', family='serif')
plt.rc('xtick', labelsize='x-large')
plt.rc('ytick', labelsize='x-large')


fig = plt.figure(figsize=(10, 6))

ax = fig.add_subplot(1, 1, 1)
#ax.set(xlim=(0.2, 10),
#           ylim=(0, 1));
ax.set_xlabel('Period (sec)',fontsize=30)
ax.set_ylabel('Amplitude of $\Theta$',fontsize=30)
ax.tick_params(axis='both', which='major', labelsize=28)





plt.plot(thetadic['period'],thetadic['theta'],color='k',ls='solid')

plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
plt.show()
#save iamge

#save iamge
#fig.savefig('PDMXRAY.eps', format='eps',bbox_inches = "tight")
fig.savefig('PDMXRAYAllLC.png', format='png',bbox_inches = "tight")
```



[ ]:

2 Phase fold

[67]:
```
bestperiod = 51732.347080
#best_freq = freq[np.argmax(PLS)]
best_freq = 1/bestperiod


summary = 'Best_frequency: {}, Period: {} days, {} Minutes'.format(best_freq,1/
 ↪best_freq,
                                                                                ␣
 ↪1/best_freq*24*60.)
sumseconds = '{} Seconds'.format( 1/best_freq*24*60.*60.)


phase = (mjd * best_freq) % 1

phase2 = phase-1
phasesall= np.concatenate((phase2, phase), axis=None)
magall = np.concatenate((mag,mag),axis=None)
mjdall =  np.concatenate((mjd,mjd),axis=None)
```

```
#colorlist = viridis(len(mjd))


p = figure(plot_width=900, plot_height=500, title=summary,active_drag='pan'
            ,␣
 ↪active_scroll='wheel_zoom',y_axis_label='flux',x_axis_label='Phase')

#p.add_layout(Title(text=sumseconds, text_font_size="10pt"), 'above')


source = ColumnDataSource(data={'phase':phasesall,
                                'flux':magall,
                                'mjd':mjdall })




#Tool to get wavelength
hover2 = HoverTool(
        tooltips=[
            ('Date', '(@mjd{0.0000})')
        ]
    )


p.add_tools(hover2)


# add a circle renderer with a size, color, and alpha
#p.add_layout(
#    Whisker(source=source, base="phase", upper="upper", lower="lower")
#)

#p.y_range.flipped = True

#mapper = LinearColorMapper(palette=colorlist, low=mjdall.min(), high=mjdall.
 ↪max())

#mapper.low_color = 'blue'
#mapper.high_color = 'red'

#color_bar = ColorBar(color_mapper=mapper, location=(0, 0.5),title='MJD')

p.circle('phase','flux',source=source )#,color={'field': 'mjd', 'transform':␣
 ↪mapper})
```

```
#p.add_layout(color_bar, 'right')
#@date{%F}'

show(p)
```

[ ]:

```
```

[ ]:

```
```

[ ]:

```
```