Advanced Astronomical Observations 2021
Session 21: Making a Photometric Lightcurve of a Variable Star

Kinoshita Daisuke

04 June 2021
publicly accessible version

About this file...

- Important information about this file
  - The author of this file is Kinoshita Daisuke.
  - The original version of this file was used for the course "Advanced Astronomical Observations" (course ID: AS6005) offered at Institute of Astronomy, National Central University from February 2021 to June 2021.
  - The file is provided in the hope that it will be useful, but there is no guarantee for the correctness. Use this file at your own risk.
  - If you are willing to use this file for your study, please feel free to use. I'll be very happy to receive feedback from you.
  - If you are willing to use this file for your teaching, please contact to Kinoshita Daisuke. When you use this file partly or entirely, please mention clearly that the author of the original version is Kinoshita Daisuke. Please help me to improve the contents of this file by sending your feedback.
  - Contact address: `https://www.instagram.com/daisuke23888/`

For this session, we try to construct a photometric lightcurve of a variable star.

1 Python scripts for this session

All the Python scripts needed for this session are available at github.com. Visit the web page `https://github.com/kinoshitadaisuke/ncu_advobs_202102` to view files. (Fig. 1)

You may try following command to download all the Python scripts for this session.

```
% git clone https://github.com/kinoshitadaisuke/ncu_advobs_202102.git
Cloning into 'ncu_advobs_202102'...
remote: Enumerating objects: 238, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (35/35), done.
Receiving objects: 100% (238/238), 3.42 MiB | 1.60 MiB/s, done.
remote: Total 238 (delta 15), reused 0 (delta 0), pack-reused 202
Resolving deltas: 100% (123/123), done.
% ls
ncu_advobs_202102/
% ls -l ncu_advobs_202102/
total 1
-rw-r--r--  1 daisuke  wheel  35149 Jun  2 04:50 LICENSE
-rw-r--r--  1 daisuke  wheel    568 Jun  2 04:50 README
-rw-r--r--  1 daisuke  wheel    375 Jun  2 04:50 REQUIREMENTS
-rw-r--r--  1 daisuke  wheel    342 Jun  2 04:50 download_python.csh
```

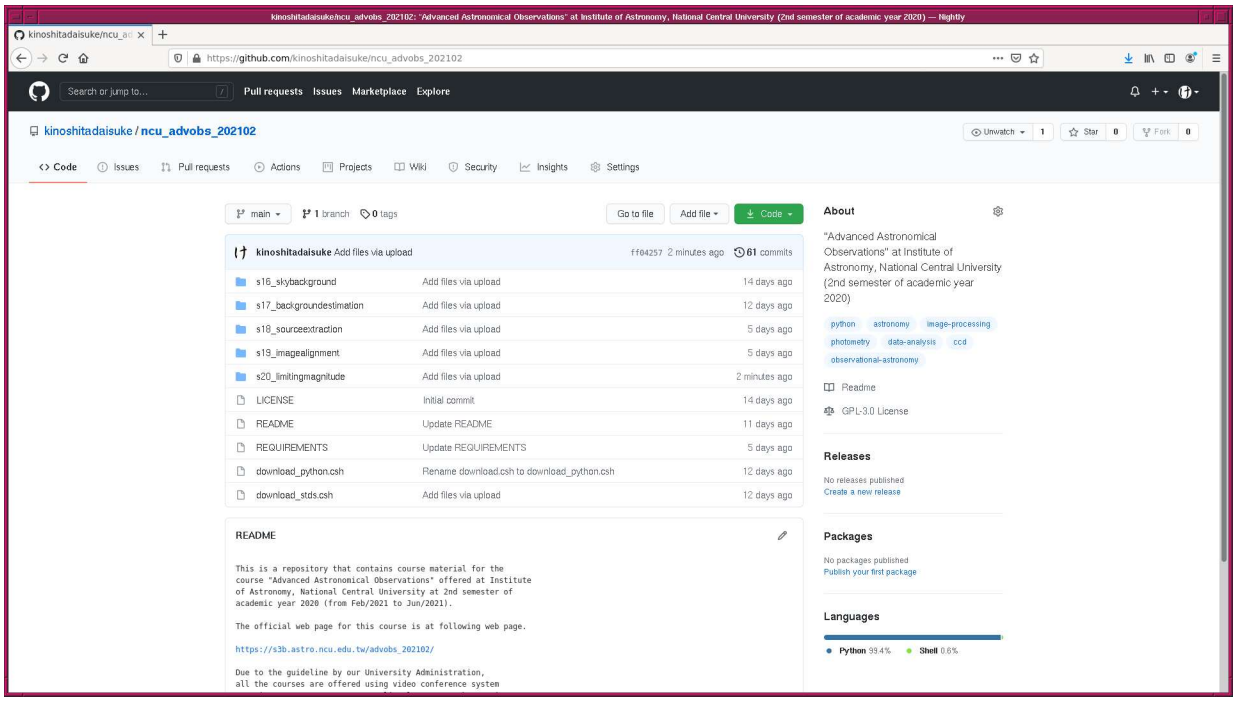

Figure 1: The GitHub repository for Python scripts for this course.

```
-rw-r--r--  1 daisuke  wheel     495 Jun  2 04:50 download_stds.csh
drwxr-xr-x  2 daisuke  wheel     512 Jun  2 04:50 s16_skybackground/
drwxr-xr-x  2 daisuke  wheel     512 Jun  2 04:50 s17_backgroundestimation/
drwxr-xr-x  2 daisuke  wheel     512 Jun  2 04:50 s18_sourceextraction/
drwxr-xr-x  2 daisuke  wheel     512 Jun  2 04:50 s19_imagealignment/
drwxr-xr-x  2 daisuke  wheel     512 Jun  2 04:50 s20_limitingmagnitude/
% ls -l ncu_advobs_202102/s20_limitingmagnitude/
total 1
-rw-r--r--  1 daisuke  wheel      75 Jun  2 04:50 README
-rw-r--r--  1 daisuke  wheel    2704 Jun  2 04:50 ao2021_s20_01.py
-rw-r--r--  1 daisuke  wheel    5031 Jun  2 04:50 ao2021_s20_02.py
-rw-r--r--  1 daisuke  wheel    4353 Jun  2 04:50 ao2021_s20_03.py
-rw-r--r--  1 daisuke  wheel    7348 Jun  2 04:50 ao2021_s20_04.py
-rw-r--r--  1 daisuke  wheel   10187 Jun  2 04:50 ao2021_s20_05.py
-rw-r--r--  1 daisuke  wheel    1127 Jun  2 04:50 ao2021_s20_06.py
-rw-r--r--  1 daisuke  wheel    2014 Jun  2 04:50 ao2021_s20_07.py
-rw-r--r--  1 daisuke  wheel   52789 Jun  2 04:50 limmag_288_289.png
% head -15 ncu_advobs_202102/s20_limitingmagnitude/ao2021_s20_01.py
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2021/06/01 17:13:11 (CST) daisuke>
#

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib
```

If above command fails on your computer, check whether or not you have the command `git`. You see something like following if you have the command `git` on your computer.

```
% which git
/usr/pkg/bin/git
```

You see following if you do not have the command `git` installed on your computer.

```
% which git
git: Command not found.
```

If you do not have the command `git`, install `git` on your computer.

2 Making a symbolic link

Make a symbolic link pointing to the directory that contains reduced data taken on 14/Feb/2021.

```
% ln -s ../../14_reduction3/script_14/ccdred_20210511_122633 0214red
% ls
0214red@
% ls -l
total 0
lrwxr-xr-x  1 daisuke  taiwan  52 Jun  3 16:43 0214red@ -> ../../14_reduction3/s
cript_14/ccdred_20210511_122633
% ls 0214red/
ccdred.log                              lot_20210214_0229_df.fits
dark_00005000.fits                      lot_20210214_0230_df.fits
dark_00010000.fits                      lot_20210214_0231_df.fits
dark_00015000.fits                      lot_20210214_0232_df.fits
dark_00045000.fits                      lot_20210214_0233_df.fits
dark_00060000.fits                      lot_20210214_0237_df.fits
dark_00180000.fits                      lot_20210214_0238_df.fits
flat_gp_Astrodon_2019.fits              lot_20210214_0239_df.fits
flat_ip_Astrodon_2019.fits              lot_20210214_0240_df.fits
flat_rp_Astrodon_2019.fits              lot_20210214_0241_df.fits

.....

lot_20210214_0213_df.fits               lot_20210214_0379_df.fits
lot_20210214_0217_df.fits               lot_20210214_0380_df.fits
lot_20210214_0218_df.fits               lot_20210214_0381_df.fits
lot_20210214_0219_df.fits               lot_20210214_0382_df.fits
lot_20210214_0220_df.fits               lot_20210214_0386_df.fits
lot_20210214_0221_df.fits               lot_20210214_0387_df.fits
lot_20210214_0225_df.fits               lot_20210214_0388_df.fits
lot_20210214_0226_df.fits               lot_20210214_0389_df.fits
lot_20210214_0227_df.fits               lot_20210214_0390_df.fits
lot_20210214_0228_df.fits
```

3 Choosing and copying FITS files

Make a Python script which works like "`gethead`" command of WCSTools. If you do not know "`gethead`" command, then visit following website and learn about "`gethead`" command.

- WCSTools

- http://tdc-www.harvard.edu/wcstools/

Python Code 1: ao2021_s21_01.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing astropy module
import astropy.io.fits

# constructing parser object
desc = "a utility like gethead of WCSTools"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-k', '--keyword', nargs='+', \
                     help='FITS keywords (e.g. "TIME-OBS EXPTIME FILTER")')
parser.add_argument ('files', nargs='+', help='FITS files')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
list_keyword = args.keyword
files_fits   = args.files

# function to read header of a FITS file
def read_fits_header (file_fits):
    # opening FITS file
    with astropy.io.fits.open (file_fits) as hdu:
        # reading header and image
        header = hdu[0].header
        # if no image in PrimaryHDU, then read next HDU
        if (header['NAXIS'] == 0):
            header = hdu[1].header
    # returning header and image
    return (header)

# printing a header
first_line = "# file name"
for keyword in list_keyword:
    first_line = first_line + ", " + keyword
print (first_line)

# processing FITS files one-by-one
for file_fits in files_fits:
    # check of file name
    if not (file_fits[-5:] == '.fits'):
        # printing a message to stderr
        print ("Input file \"%s\" is not a FITS file." % file_fits, \
               file=sys.stderr)
        print ("Input file must be \"*.fits\".")
```

```
        # skipping
        continue

    # making pathlib object
    path_fits = pathlib.Path (file_fits)
    filename  = path_fits.name

    # if the file does not exist, then skip
    if not ( path_fits.exists () and path_fits.is_file () ):
        # printing a message to stderr
        print ("The file \"%s\" does not exist or is not a regular file." \
               % file_fits, file=sys.stderr)
        # skipping
        continue

    # reading FITS header
    header = read_fits_header (file_fits)

    # making an empty list for storing values of keywords
    list_values = []

    # values of keywords
    for keyword in list_keyword:
        # if the keyword exists, then read the value from FITS header
        if (keyword in header):
            value = header[keyword]
        # if the keyword does not exists, then use '__NONE__' as a value
        else:
            value = '__NONE__'
        # appending the value to the list
        list_values.append (value)

    # printing information
    data = str (filename)
    for value in list_values:
        data = data + " " + str (value)
    print (data)
```

Execute the script.

```
% chmod a+x ao2021_s21_01.py
% ./ao2021_s21_01.py -h
usage: ao2021_s21_01.py [-h] [-k KEYWORD [KEYWORD ...]] files [files ...]

a utility like gethead of WCSTools

positional arguments:
  files                 FITS files

optional arguments:
  -h, --help            show this help message and exit
  -k KEYWORD [KEYWORD ...], --keyword KEYWORD [KEYWORD ...]
                        FITS keywords (e.g. "TIME-OBS EXPTIME FILTER")

% ./ao2021_s21_01.py 0214red/*.fits -k time-obs imagetyp exptime filter object
# file name, time-obs, imagetyp, exptime, filter, object
dark_00005000.fits 22:27:01 DARK 5.0 __NONE__ domeflat
dark_00010000.fits 22:32:02 DARK 10.0 __NONE__ domeflat
```

```
dark_00015000.fits 22:38:46 DARK 15.0 __NONE__ domeflat
dark_00045000.fits 23:10:34 DARK 45.0 __NONE__ domeflat
dark_00060000.fits 23:28:55 DARK 60.0 __NONE__ domeflat
dark_00180000.fits 01:09:05 DARK 180.0 __NONE__ domeflat
flat_gp_Astrodon_2019.fits 21:35:45 FLAT 45.0 gp_Astrodon_2019 dark
flat_ip_Astrodon_2019.fits 20:43:23 FLAT 60.0 ip_Astrodon_2019 dark
flat_rp_Astrodon_2019.fits 21:37:34 FLAT 45.0 rp_Astrodon_2019 dark
lot_20210214_0085_df.fits 16:23:10 LIGHT 60.0 rp_Astrodon_2019 V0678VIR

.....

lot_20210214_0378_df.fits 21:06:31 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0379_df.fits 21:07:47 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0380_df.fits 21:09:06 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0381_df.fits 21:10:22 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0382_df.fits 21:11:38 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0386_df.fits 21:14:26 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0387_df.fits 21:15:41 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0388_df.fits 21:16:56 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0389_df.fits 21:18:10 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0390_df.fits 21:19:24 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
```

For this session, we process the data of the variable star V0678 Vir. List FITS files of V0678 Vir data.

```
% ./ao2021_s21_01.py 0214red/*.fits -k time-obs imagetyp exptime filter object \
? | grep V0678VIR
lot_20210214_0085_df.fits 16:23:10 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0086_df.fits 16:24:26 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0087_df.fits 16:25:40 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0088_df.fits 16:26:54 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0089_df.fits 16:28:09 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0117_df.fits 16:53:17 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0118_df.fits 16:54:35 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0119_df.fits 16:55:52 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0120_df.fits 16:57:10 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0121_df.fits 16:58:27 LIGHT 60.0 rp_Astrodon_2019 V0678VIR

.....

lot_20210214_0378_df.fits 21:06:31 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0379_df.fits 21:07:47 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0380_df.fits 21:09:06 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0381_df.fits 21:10:22 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0382_df.fits 21:11:38 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0386_df.fits 21:14:26 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0387_df.fits 21:15:41 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0388_df.fits 21:16:56 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0389_df.fits 21:18:10 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
lot_20210214_0390_df.fits 21:19:24 LIGHT 60.0 rp_Astrodon_2019 V0678VIR
% ./ao2021_s21_01.py 0214red/*.fits -k time-obs imagetyp exptime filter object \
? | grep V0678VIR | wc
      81     486    5832
```

81 files are found.
Check whether or not all the data are acquired using r'-band filter.

```
% ./ao2021_s21_01.py 0214red/*.fits -k time-obs imagetyp exptime filter object \
? | grep V0678VIR | grep -v rp_Astrodon_2019
lot_20210214_0337_df.fits 20:25:06 LIGHT 60.0 gp_Astrodon_2019 V0678VIR
lot_20210214_0338_df.fits 20:26:22 LIGHT 60.0 gp_Astrodon_2019 V0678VIR
lot_20210214_0341_df.fits 20:30:20 LIGHT 60.0 ip_Astrodon_2019 V0678VIR
lot_20210214_0342_df.fits 20:31:34 LIGHT 60.0 ip_Astrodon_2019 V0678VIR
```

There are two g'-band data and two more i'-band data.

```
% ./ao2021_s21_01.py 0214red/*.fits -k time-obs imagetyp exptime filter object \
? | grep V0678VIR | grep rp_Astrodon_2019 | wc
      77     462    5544
```

Number of r'-band data of V0678 Vir is 77.

4 Copying FITS files

Make a Python script to copy FITS files of V0678 Vir data.

Python Code 2: ao2021_s21_02.py

```
#!/usr/pkg/bin/python3.9

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing shutil module
import shutil

# source directory
dir_src = '0214red'

# destination directory
dir_dst = 'v0678vir'

# making a directory if it does not exist
path_dst = pathlib.Path (dir_dst)
path_dst.mkdir (mode=0o755, exist_ok=True)

# reading data from standard input
for line in sys.stdin:
    # if the line starts with '#", then skip
    if (line[0] == '#'):
        continue
    # splitting data
    record = line.split ()
    # data
    filename   = record[0]
    imagetype  = record[1]
    exptime    = float (record[2])
    filtername = record[3]
    objectname = record[4]
    # rejecting data if criteria do not match
```

```
    if not ( (imagetype == 'LIGHT') and (exptime == 60.0) \
             and (filtername == 'rp_Astrodon_2019') \
             and (objectname == 'V0678VIR') ):
        continue
    # copying file
    file_fits = "%s/%s" % (dir_src, filename)
    path_fits = pathlib.Path (file_fits)
    print ("copying file from %s to %s..." % (path_fits, dir_dst) )
    shutil.copy2 (path_fits, dir_dst)
```

Run the script and copy FITS files.

```
% chmod a+x ao2021_s21_02.py
% ./ao2021_s21_01.py 0214red/*.fits -k imagetyp exptime filter object | \
? ./ao2021_s21_02.py
copying file from 0214red/lot_20210214_0085_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0086_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0087_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0088_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0089_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0117_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0118_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0119_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0120_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0121_df.fits to v0678vir...

.....

copying file from 0214red/lot_20210214_0378_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0379_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0380_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0381_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0382_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0386_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0387_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0388_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0389_df.fits to v0678vir...
copying file from 0214red/lot_20210214_0390_df.fits to v0678vir...
% ls
0214red@            ao2021_s21_01.py~*  ao2021_s21_02.py~   test_argparse.py~
ao2021_s21_01.py*   ao2021_s21_02.py*   test_argparse.py*   v0678vir/
% ls v0678vir/
lot_20210214_0085_df.fits               lot_20210214_0241_df.fits
lot_20210214_0086_df.fits               lot_20210214_0257_df.fits
lot_20210214_0087_df.fits               lot_20210214_0258_df.fits
lot_20210214_0088_df.fits               lot_20210214_0259_df.fits
lot_20210214_0089_df.fits               lot_20210214_0260_df.fits
lot_20210214_0117_df.fits               lot_20210214_0261_df.fits
lot_20210214_0118_df.fits               lot_20210214_0277_df.fits
lot_20210214_0119_df.fits               lot_20210214_0278_df.fits
lot_20210214_0120_df.fits               lot_20210214_0279_df.fits
lot_20210214_0121_df.fits               lot_20210214_0280_df.fits

.....

lot_20210214_0198_df.fits               lot_20210214_0379_df.fits
lot_20210214_0217_df.fits               lot_20210214_0380_df.fits
lot_20210214_0218_df.fits               lot_20210214_0381_df.fits
```

```
lot_20210214_0219_df.fits                lot_20210214_0382_df.fits
lot_20210214_0220_df.fits                lot_20210214_0386_df.fits
lot_20210214_0221_df.fits                lot_20210214_0387_df.fits
lot_20210214_0237_df.fits                lot_20210214_0388_df.fits
lot_20210214_0238_df.fits                lot_20210214_0389_df.fits
lot_20210214_0239_df.fits                lot_20210214_0390_df.fits
lot_20210214_0240_df.fits
% ls v0678vir/*.fits | wc
      77      77    2695
```

5 Visual inspection of a FITS file

Use Ginga to check the image. (Fig. 2)

```
% ginga v0678vir/lot_20210214_0117_df.fits
```

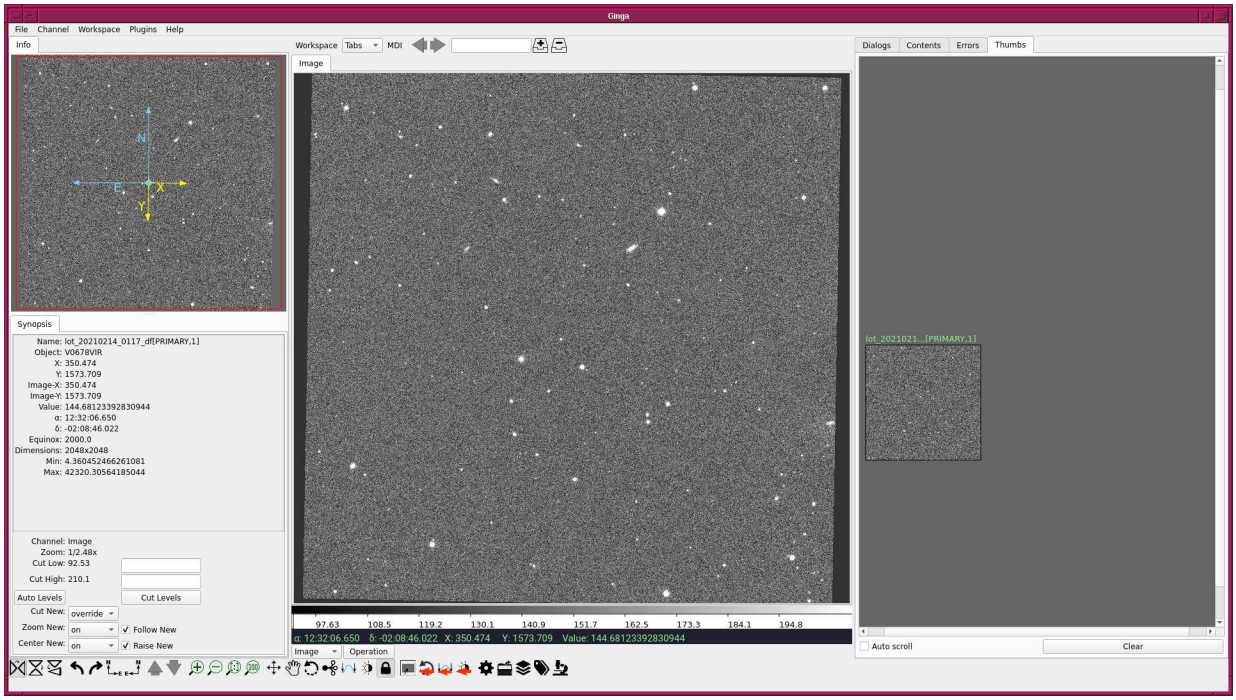

Figure 2: Image of `lot_20210214_0117_df.fits`.

6 General Catalog of Variable Stars

Visit the website of "General Catalog of Variable Stars" (GCVS).

- General Catalog of Variable Stars (GCVS)
  - `http://www.sai.msu.su/gcvs/gcvs/`

Download GCVS version 5.1.

```
% curl -o gcvs5.data http://www.sai.msu.su/gcvs/gcvs/gcvs5/gcvs5.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
```

```
100 12.9M  100 12.9M    0     0   412k      0  0:00:32  0:00:32 --:--:--  513k
% ls -l gcvs5.data
-rw-r--r--  1 daisuke  taiwan  13608033 Jun  3 19:05 gcvs5.data
```

Search for the object “V0678 Vir”.

```
% grep 'V0678 Vir' gcvs5.data
860678 |V0678 Vir  |123148.11 -020602.3 |EW        | 14.74   |  15.13     |  15.
12     |CV|56282.005   |     |     0.2266085      |     |                  |82001
 GSC  |           |-0.016 -0.004|         | |UCAC     |          |V0678 Vir |
```

RA and Dec of V0678 Vir are 12h31m48.11s and -02d06m02.3s, respectively.

7 Identifying V0678 Vir on an image

Make a Python script to identify V0678 Vir on an image.

Python Code 3: ao2021_s21_03.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.coordinates
import astropy.visualization
import astropy.wcs

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# list of colours
list_colour = ['maroon', 'red', 'coral', 'bisque', 'orange', \
               'wheat', 'yellow', 'green', 'lime', 'aqua', \
               'skyblue', 'blue', 'indigo', 'violet', 'pink']

# list of markers
list_marker = ['o', 'v', '^', 's', 'p', 'h', '8']

# list of cmap
list_cmap = [ 'viridis', 'plasma', 'inferno', 'magma', 'cividis', 'gray', \
              'bone', 'pink', 'spring', 'summer', 'autumn', 'winter', \
              'cool', 'hot', 'copper', 'hsv', 'ocean', 'terrain', 'gnuplot', \
              'rainbow', 'turbo'
            ]

# constructing parser object
desc = "marking an object"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-i', '--file-input', default='', \
```

```
                      help='input FITS file')
parser.add_argument ('-o', '--file-output', default='', \
                      help='output graphics file')
parser.add_argument ('-p', '--radec', default='', \
                      help='RA and Dec in hh:mm:ss.ss,+/-dd:mm:ss.s format')
parser.add_argument ('-c', '--colour', choices=list_colour, default='red', \
                      help='choice of colour (default: red)')
parser.add_argument ('-m', '--marker', choices=list_marker, default='o', \
                      help='choice of marker (default: o)')
parser.add_argument ('-a', '--cmap', choices=list_cmap, default='bone', \
                      help='choice of cmap (default: bone)')
parser.add_argument ('-s', '--size', type=float, default=10.0, \
                      help='size of marker (default: 10)')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_input  = args.file_input
file_output = args.file_output
coord_radec = args.radec
colour      = args.colour
marker      = args.marker
cmap        = args.cmap
size        = args.size

# check of input file
if not (file_input[-5:] == '.fits'):
    print ("Input file must be a FITS file.", file=sys.stderr)
    sys.exit ()

# check of output file
if not ( (file_output[-4:] == '.eps') or (file_output[-4:] == '.pdf') \
         or (file_output[-4:] == '.png') or (file_output[-3:] == '.ps')):
    print ("Output file must be either EPS, PDF, PNG, or PS file.")
    sys.exit ()

# RA and Dec
coord_ra, coord_dec = coord_radec.split (',')

# check of RA value
if not (':' in coord_ra):
    print ("RA must be hh:mm:ss.ss format.", file=sys.stderr)
    sys.exit ()

# check of Dec value
if not (':' in coord_dec):
    print ("Dec must be +/-dd:mm:ss.ss format.", file=sys.stderr)
    sys.exit ()

# printing input parameters
print ("#")
print ("# Input parameters")
print ("#")
print ("#   input FITS file  = %s" % file_input)
print ("#   output FITS file = %s" % file_output)
print ("#   coordinate       = %s" % coord_radec)
print ("#   colour of marker = %s" % colour)
print ("#   shape of marker  = %s" % marker)
```

```
print ("#   size of marker    = %f" % size)
print ("#   cmap              = %s" % cmap)
print ("#")

# function to read a FITS file
def read_fits (file_fits):
    # reading FITS file
    with astropy.io.fits.open (file_fits) as hdu:
        # reading header and image
        header = hdu[0].header
        image  = hdu[0].data
        # if no image in PrimaryHDU, then read next HDU
        if (header['NAXIS'] == 0):
            header = hdu[1].header
            image  = hdu[1].data
    # returning header and image
    return (header, image)

# units
u_ha  = astropy.units.hourangle
u_deg = astropy.units.deg

# coordinate
coord = astropy.coordinates.SkyCoord (coord_ra, coord_dec, unit=(u_ha, u_deg) )

# reading FITS file
(header, image) = read_fits (file_input)

# WCS
wcs = astropy.wcs.WCS (header)

print ("#")
print ("# WCS information from FITS file")
print ("#")
print ("# CD matrix")
print ("#  [")
print ("#    [%g %g]" \
       % (wcs.pixel_scale_matrix[0][0], wcs.pixel_scale_matrix[0][1]) )
print ("#    [%g %g]" \
       % (wcs.pixel_scale_matrix[1][0], wcs.pixel_scale_matrix[1][1]) )
print ("#  ]")

# check of WCS information
if ( (wcs.pixel_scale_matrix[0][0] == 1) \
     and (wcs.pixel_scale_matrix[0][1] == 0) \
     and (wcs.pixel_scale_matrix[1][0] == 0) \
     and (wcs.pixel_scale_matrix[1][1] == 1) \
    ):
    print ("No WCS information in FITS file.", file=sys.stderr)
    sys.exit ()

# conversion from sky coordinate into pixel coordinate
(x, y) = wcs.world_to_pixel (coord)

# printing coordinate information
print ("#")
print ("# coordinate")
print ("#")
print ("#   (RA, Dec) = (%s, %s)" % (coord.ra, coord.dec) )
```

```
print ("#      ==> (x, y) = (%f, %f)" % (x, y) )
print ("#")

# making objects "fig" and "ax"
fig = matplotlib.figure.Figure ()
matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax = fig.add_subplot (111)

# plotting image
norm \
    = astropy.visualization.mpl_normalize.ImageNormalize \
    ( stretch=astropy.visualization.HistEqStretch (image) )
im = ax.imshow (image, cmap=cmap, norm=norm, origin='upper')
ax.plot (x, y, marker=marker, color=colour, markersize=size, fillstyle='none')
ax.set_xlabel ('X [pixel]')
ax.set_ylabel ('Y [pixel]')

# saving to file
fig.savefig (file_output, dpi=225)
```

Execute the script, and show the location of V0678 Vir.

```
% chmod a+x ao2021\_s21\_03.py
% ./ao2021_s21_03.py -h
usage: ao2021_s21_03.py [-h] [-i FILE_INPUT] [-o FILE_OUTPUT] [-p RADEC]
                        [-c {maroon,red,coral,bisque,orange,wheat,yellow,green,l
ime,aqua,skyblue,blue,indigo,violet,pink}]
                        [-m {o,v,^,s,p,h,8}]
                        [-a {viridis,plasma,inferno,magma,cividis,gray,bone,pink
,spring,summer,autumn,winter,cool,hot,copper,hsv,ocean,terrain,gnuplot,rainbow,t
urbo}]
                        [-s SIZE]

marking an object

optional arguments:
  -h, --help            show this help message and exit
  -i FILE_INPUT, --file-input FILE_INPUT
                        input FITS file
  -o FILE_OUTPUT, --file-output FILE_OUTPUT
                        output graphics file
  -p RADEC, --radec RADEC
                        RA and Dec in hh:mm:ss.ss,+/-dd:mm:ss.s format
  -c {maroon,red,coral,bisque,orange,wheat,yellow,green,lime,aqua,skyblue,blue,i
ndigo,violet,pink}, --colour {maroon,red,coral,bisque,orange,wheat,yellow,green,
lime,aqua,skyblue,blue,indigo,violet,pink}
                        choice of colour (default: red)
  -m {o,v,^,s,p,h,8}, --marker {o,v,^,s,p,h,8}
                        choice of marker (default: o)
  -a {viridis,plasma,inferno,magma,cividis,gray,bone,pink,spring,summer,autumn,w
inter,cool,hot,copper,hsv,ocean,terrain,gnuplot,rainbow,turbo}, --cmap {viridis,
plasma,inferno,magma,cividis,gray,bone,pink,spring,summer,autumn,winter,cool,hot
,copper,hsv,ocean,terrain,gnuplot,rainbow,turbo}
                        choice of cmap (default: bone)
  -s SIZE, --size SIZE  size of marker (default: 10)

% ./ao2021_s21_03.py -i v0678vir/lot_20210214_0117_df.fits \
? -o v0678vir_0117.pdf -p 12:31:48.11,-02:06:02.3
```

```
#
# Input parameters
#
#   input FITS file  = v0678vir/lot_20210214_0117_df.fits
#   output FITS file = v0678vir_0117.pdf
#   coordinate       = 12:31:48.11,-02:06:02.3
#   colour of marker = red
#   shape of marker  = o
#   size of marker   = 10.000000
#   cmap             = bone
#
WARNING: FITSFixedWarning: RADECSYS= 'FK5 ' / Equatorial coordinate system
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
#
# WCS information from FITS file
#
# CD matrix
#  [
#    [-0.000106958 1.86366e-06]
#    [-1.86446e-06 -0.000106912]
#  ]
#
# coordinate
#
#   (RA, Dec) = (187d57m01.65s, -2d06m02.3s)
#     ==> (x, y) = (1063.579512, 1134.858633)
#
% ls -l v0678vir_0117.pdf
-rw-r--r--  1 daisuke  taiwan  1854211 Jun  3 20:55 v0678vir_0117.pdf
```

Use the command `xpdf` to show the PDF file. (Fig. 3)

```
% xpdf v0678vir_0117.pdf
```

8 Source extraction

Make a Python script to carry out source extraction.

Python Code 4: ao2021_s21_04.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing sys module
import sys

# importing datetime module
import datetime

# importing numpy module
import numpy.ma

# importing astropy module
import astropy.convolution
```

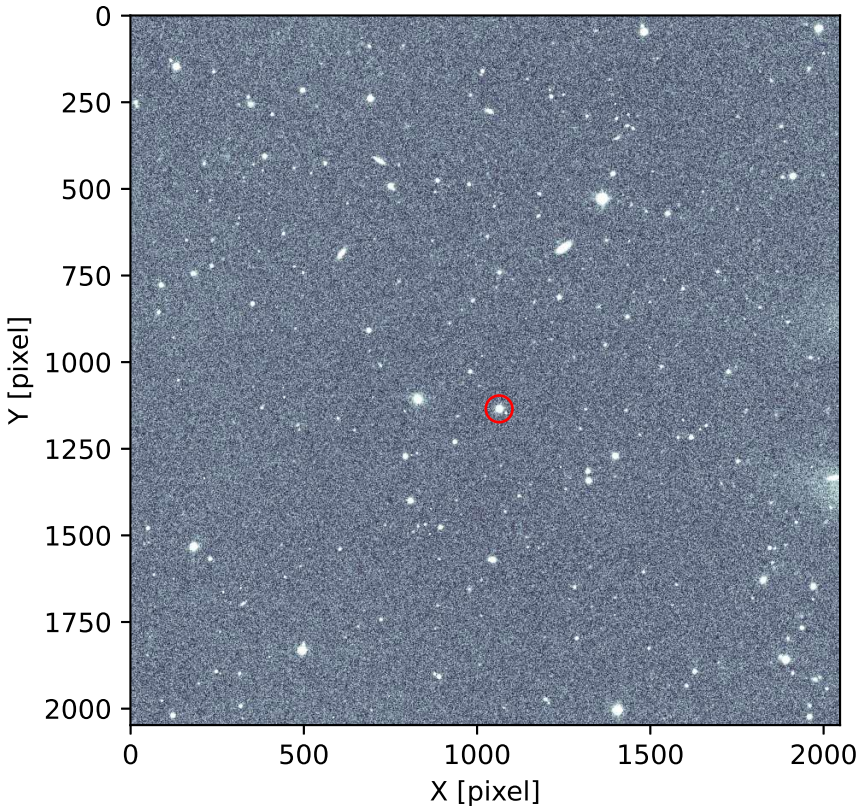

Figure 3: The location of V0678 Vir on the image `lot_20210214_0117_df.fits`.

```
import astropy.io.fits
import astropy.io.votable
import astropy.stats

# importing photutils module
import photutils.segmentation

# date/time
now = datetime.datetime.now ()

# constructing parser object
desc   = 'source extraction using image segmentation and deblending'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-i', '--file-input', default='', \
                     help='input FITS file name')
parser.add_argument ('-o', '--file-output', default='', \
                     help='output VOTable file name')
parser.add_argument ('-t', '--threshold', type=float, default=10.0, \
                     help='detection threshold in sigma (default: 10)')
parser.add_argument ('-u', '--threshold-for-sky', type=float, default=2.0, \
                     help='detection threshold for sky estimate (default: 2)')
parser.add_argument ('-n', '--npixels', type=int, default=5, \
                     help='minimum number of pixels for detection (default: 5)')
parser.add_argument ('-s', '--dilate-size', type=int, default=21, \
                     help='dilate size (default: 21)')
parser.add_argument ('-m', '--maxiters', type=int, default=30, \
                     help='maximum number of iterations (default: 30)')
parser.add_argument ('-r', '--sigma-clipping', type=float, default=4.0, \
                     help='sigma-clipping threshold in sigma (default: 4)')
parser.add_argument ('-k', '--gaussian-fwhm', type=float, default=3.0, \
                     help='Gaussian FWHM in pixel for convolution (default: 3)')
parser.add_argument ('-a', '--kernel-size', type=int, default=3, \
                     help='Gaussian kernel array size in pixel (default: 3)')

# command-line argument analysis
args = parser.parse_args ()

# file names
file_input  = args.file_input
file_output = args.file_output

# input parameters
threshold         = args.threshold
threshold_for_sky = args.threshold
npixels           = args.npixels
dilate_size       = args.dilate_size
maxiters          = args.maxiters
rejection         = args.sigma_clipping
gaussian_fwhm     = args.gaussian_fwhm
kernel_array_size = args.kernel_size

# check of input file name
if not (file_input[-5:] == '.fits'):
    print ("Input file must be a FITS file.")
    sys.exit ()

# check of catalogue file name
```

```
if not (file_output[-4:] == '.vot'):
    print ("Output file must be a VOTable file.")
    sys.exit ()

# opening FITS file
with astropy.io.fits.open (file_input) as hdu:
    # reading header and image
    header = hdu[0].header
    image  = hdu[0].data
    # if no image in PrimaryHDU, then read next HDU
    if (header['NAXIS'] == 0):
        header = hdu[1].header
        image  = hdu[1].data

# making source mask
source_mask \
    = photutils.segmentation.make_source_mask (image, threshold_for_sky,
                                               npixels=npixels,
                                               sigclip_iters=maxiters,
                                               dilate_size=dilate_size)

# making masked array
image_masked = numpy.ma.array (image, mask=source_mask)

# sigma-clipping
skybg_mean, skybg_median, skybg_stddev \
    = astropy.stats.sigma_clipped_stats (image, sigma=rejection)

# mode calculation using empirical formula
skybg_mode = 3.0 * skybg_median - 2.0 * skybg_mean

# detection threshold in ADU
threshold_adu = skybg_mode + threshold * skybg_stddev

# 2D Gaussian kernel for convolution
gaussian_sigma = gaussian_fwhm * astropy.stats.gaussian_fwhm_to_sigma
kernel = astropy.convolution.Gaussian2DKernel (gaussian_sigma, \
                                               x_size=kernel_array_size, \
                                               y_size=kernel_array_size)
kernel.normalize ()

# source detection
image_segm = photutils.segmentation.detect_sources (image, threshold_adu, \
                                                    npixels=npixels, \
                                                    filter_kernel=kernel)

# deblending
image_deblend = photutils.segmentation.deblend_sources (image, image_segm, \
                                                        npixels=npixels, \
                                                        filter_kernel=kernel, \
                                                        nlevels=32, \
                                                        contrast=0.001)

# making a source catalogue
catalogue = photutils.segmentation.SourceCatalog (image, image_deblend)

# making a table
table_sources = catalogue.to_table ()
```

```
# converting a table into a VOTable
votable_sources = astropy.io.votable.from_table (table_sources)

# writing VOTable to a file
astropy.io.votable.writeto (votable_sources, file_output)
```

Run the script, and find stars.

```
% chmod a+x ao2021_s21_04.py
% ./ao2021_s21_04.py -h
usage: ao2021_s21_04.py [-h] [-i FILE_INPUT] [-o FILE_OUTPUT] [-t THRESHOLD]
                        [-u THRESHOLD_FOR_SKY] [-n NPIXELS] [-s DILATE_SIZE]
                        [-m MAXITERS] [-r SIGMA_CLIPPING] [-k GAUSSIAN_FWHM]
                        [-a KERNEL_SIZE]

source extraction using image segmentation and deblending

optional arguments:
  -h, --help            show this help message and exit
  -i FILE_INPUT, --file-input FILE_INPUT
                        input FITS file name
  -o FILE_OUTPUT, --file-output FILE_OUTPUT
                        output VOTable file name
  -t THRESHOLD, --threshold THRESHOLD
                        detection threshold in sigma (default: 10)
  -u THRESHOLD_FOR_SKY, --threshold-for-sky THRESHOLD_FOR_SKY
                        detection threshold for sky estimate (default: 2)
  -n NPIXELS, --npixels NPIXELS
                        minimum number of pixels for detection (default: 5)
  -s DILATE_SIZE, --dilate-size DILATE_SIZE
                        dilate size (default: 21)
  -m MAXITERS, --maxiters MAXITERS
                        maximum number of iterations (default: 30)
  -r SIGMA_CLIPPING, --sigma-clipping SIGMA_CLIPPING
                        sigma-clipping threshold in sigma (default: 4)
  -k GAUSSIAN_FWHM, --gaussian-fwhm GAUSSIAN_FWHM
                        Gaussian FWHM in pixel for convolution (default: 3)
  -a KERNEL_SIZE, --kernel-size KERNEL_SIZE
                        Gaussian kernel array size in pixel (default: 3)

% ./ao2021_s21_04.py -t 50 -i v0678vir/lot_20210214_0117_df.fits -o 0117.vot
% ls -l 0117.vot
-rw-r--r--  1 daisuke  taiwan  16533 Jun  3 22:03 0117.vot
```

Make a Python script to read a VOTable file and plot locations of extracted sources.

Python Code 5: ao2021_s21_05.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing sys module
import sys

# importing numpy module
import numpy
```

```
# importing astropy module
import astropy.io.fits
import astropy.io.votable
import astropy.visualization
import astropy.wcs

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# constructing parser object
desc = "Reading a VOTable file and plotting locations of extracted sources"
parser = argparse.ArgumentParser (description=desc)

# list of colours
list_colours = ['maroon', 'red', 'coral', 'bisque', 'orange', \
                'wheat', 'yellow', 'green', 'lime', 'aqua', \
                'skyblue', 'blue', 'indigo', 'violet', 'pink']

# list of markers
list_markers = ['o', 'v', '^', 's', 'p', 'h', '8']

# list of cmap
list_cmap = [ 'viridis', 'plasma', 'inferno', 'magma', 'cividis', 'gray', \
              'bone', 'pink', 'spring', 'summer', 'autumn', 'winter', \
              'cool', 'hot', 'copper', 'hsv', 'ocean', 'terrain', 'gnuplot', \
              'rainbow', 'turbo'
            ]

# adding arguments
parser.add_argument ('-f', '--file-fits', default='', \
                     help='input FITS file (*.fits)')
parser.add_argument ('-t', '--file-votable', default='', \
                     help='input VOTable file (*.vot)')
parser.add_argument ('-o', '--file-output', default='', \
                     help='output graphics file (EPS, PDF, PNG, or PS file)')
parser.add_argument ('-b', '--cmap', choices=list_cmap, default='bone', \
                     help='cmap for colour bar (default: bone)')
parser.add_argument ('-c', '--colour', choices=list_colours, default='red', \
                     help='colour of marker (default: red)')
parser.add_argument ('-m', '--marker', choices=list_markers, default='o', \
                     help='shape of marker (default: o)')
parser.add_argument ('-r', '--radius', type=float, default=10.0, \
                     help='radius of marker in pixel (default: 10)')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_fits    = args.file_fits
file_votable = args.file_votable
file_output  = args.file_output
cmap         = args.cmap
colour       = args.colour
marker       = args.marker
radius       = args.radius

# check of input file
```

```
if not (file_fits[-5:] == '.fits'):
    print ("Input file given by -f option must be a FITS file.")
    sys.exit ()
if not (file_votable[-4:] == '.vot'):
    print ("Input file given by -t option must be a VOTable file.")
    sys.exit ()

# check of output file
if not ( (file_output[-4:] == '.eps') or (file_output[-4:] == '.pdf') \
         or (file_output[-4:] == '.png') or (file_output[-3:] == '.ps') ):
    print ("Output file must be either EPS, PDF, PNG, or PS file.")
    sys.exit ()

# function to read header and image of a FITS file
def read_fits (file_fits):
    # opening FITS file
    with astropy.io.fits.open (file_fits) as hdu:
        # reading header and image
        header = hdu[0].header
        image  = hdu[0].data
        # if no image in PrimaryHDU, then read next HDU
        if (header['NAXIS'] == 0):
            header = hdu[1].header
            image  = hdu[1].data
    # returning header and image
    return (header, image)

# reading VOTable file
votable_sources = astropy.io.votable.parse (file_votable)

# reading the first table in VOTable
table_sources = votable_sources.get_first_table ()

# X and Y coordinates of sources
centroid_x = table_sources.array['xcentroid']
centroid_y = table_sources.array['ycentroid']

# reading FITS file
header, image = read_fits (file_fits)
wcs           = astropy.wcs.WCS (header)

# making objects "fig" and "ax"
fig = matplotlib.figure.Figure ()
matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax = fig.add_subplot (111)

# plotting image
norm \
    = astropy.visualization.mpl_normalize.ImageNormalize \
    ( stretch=astropy.visualization.HistEqStretch (image) )
im = ax.imshow (image, cmap=cmap, norm=norm, origin='upper')
ax.plot (centroid_x, centroid_y, marker=marker, color=colour, \
         markersize=radius, fillstyle='none', linestyle='None')
ax.set_xlabel ('X [pixel]')
ax.set_ylabel ('Y [pixel]')

# saving to file
fig.savefig (file_output, dpi=225)
```

Run the script, and show locations of extracted sources.

```
% chmod a+x ao2021_s21_05.py
% ./ao2021_s21_05.py -h
usage: ao2021_s21_05.py [-h] [-f FILE_FITS] [-t FILE_VOTABLE] [-o FILE_OUTPUT]
                        [-b {viridis,plasma,inferno,magma,cividis,gray,bone,pink
,spring,summer,autumn,winter,cool,hot,copper,hsv,ocean,terrain,gnuplot,rainbow,t
urbo}]
                        [-c {maroon,red,coral,bisque,orange,wheat,yellow,green,l
ime,aqua,skyblue,blue,indigo,violet,pink}]
                        [-m {o,v,^,s,p,h,8}] [-r RADIUS]

Reading a VOTable file and plotting locations of extracted sources

optional arguments:
  -h, --help            show this help message and exit
  -f FILE_FITS, --file-fits FILE_FITS
                        input FITS file (*.fits)
  -t FILE_VOTABLE, --file-votable FILE_VOTABLE
                        input VOTable file (*.vot)
  -o FILE_OUTPUT, --file-output FILE_OUTPUT
                        output graphics file (EPS, PDF, PNG, or PS file)
  -b {viridis,plasma,inferno,magma,cividis,gray,bone,pink,spring,summer,autumn,w
inter,cool,hot,copper,hsv,ocean,terrain,gnuplot,rainbow,turbo}, --cmap {viridis,
plasma,inferno,magma,cividis,gray,bone,pink,spring,summer,autumn,winter,cool,hot
,copper,hsv,ocean,terrain,gnuplot,rainbow,turbo}
                        cmap for colour bar (default: bone)
  -c {maroon,red,coral,bisque,orange,wheat,yellow,green,lime,aqua,skyblue,blue,i
ndigo,violet,pink}, --colour {maroon,red,coral,bisque,orange,wheat,yellow,green,
lime,aqua,skyblue,blue,indigo,violet,pink}
                        colour of marker (default: red)
  -m {o,v,^,s,p,h,8}, --marker {o,v,^,s,p,h,8}
                        shape of marker (default: o)
  -r RADIUS, --radius RADIUS
                        radius of marker in pixel (default: 10)

% ./ao2021_s21_05.py -f v0678vir/lot_20210214_0117_df.fits -t 0117.vot \
? -o 0117_sources.pdf
WARNING: FITSFixedWarning: RADECSYS= 'FK5 ' / Equatorial coordinate system
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
% ls -l 0117_sources.pdf
-rw-r--r--  1 daisuke  taiwan  1857806 Jun  3 22:15 0117_sources.pdf
```

Use the command `xpdf` to show the PDF file. (Fig. 4)

```
% xpdf 0117_sources.pdf
```

Make a Python script to check information of extracted sources, select stars suitable as reference star, and print information of those stars.

Python Code 6: ao2021_s21_06.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse
```

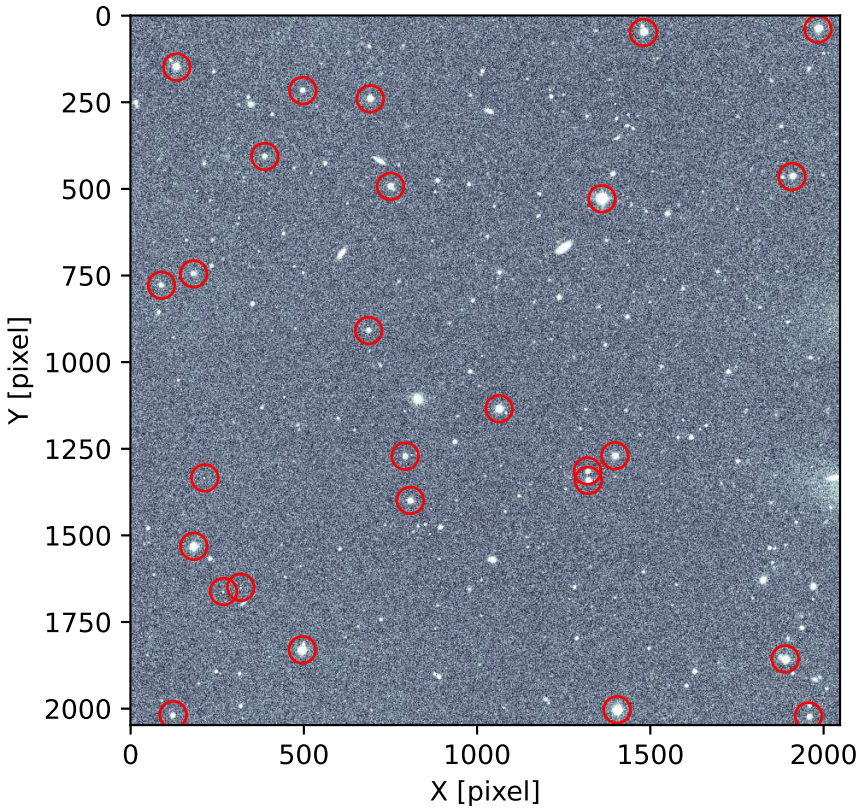

Figure 4: Locations of extracted sources.

```
# importing sys module
import sys

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits
import astropy.io.votable
import astropy.visualization
import astropy.wcs

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# constructing parser object
desc = "Reading a VOTable file and printing information of extracted sources"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-f', '--file-fits', default='', \
                     help='input FITS file (*.fits)')
parser.add_argument ('-t', '--file-votable', default='', \
                     help='input VOTable file (*.vot)')
parser.add_argument ('-o', '--file-output', default='', \
                     help='output file')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_fits    = args.file_fits
file_votable = args.file_votable
file_output  = args.file_output

# check of input file
if not (file_fits[-5:] == '.fits'):
    print ("Input file given by -f option must be a FITS file.")
    sys.exit ()
if not (file_votable[-4:] == '.vot'):
    print ("Input file given by -t option must be a VOTable file.")
    sys.exit ()

# check of output file
if (file_output == ''):
    print ("Output file name must be specified.")
    sys.exit ()

# function to read header and image of a FITS file
def read_fits (file_fits):
    # opening FITS file
    with astropy.io.fits.open (file_fits) as hdu:
        # reading header and image
        header = hdu[0].header
        image  = hdu[0].data
        # if no image in PrimaryHDU, then read next HDU
        if (header['NAXIS'] == 0):
            header = hdu[1].header
```

```
            image  = hdu[1].data
    # returning header and image
    return (header, image)

# reading VOTable file
votable_sources = astropy.io.votable.parse (file_votable)

# reading the first table in VOTable
table_sources = votable_sources.get_first_table ()

# X and Y coordinates of sources
centroid_x   = table_sources.array['xcentroid']
centroid_y   = table_sources.array['ycentroid']
eccentricity = table_sources.array['eccentricity']
kron_flux    = table_sources.array['kron_flux']
max_value    = table_sources.array['max_value']

# reading FITS file
header, image = read_fits (file_fits)
wcs           = astropy.wcs.WCS (header)

# making an empty dictionary
stars = {}

# construction of a multi-dimensional dictionary
for i in range ( len (centroid_x) ):
    # star ID
    id = "%02d" % i
    # adding data to the dictionary
    stars[id]          = {}
    stars[id]['x']     = centroid_x[i]
    stars[id]['y']     = centroid_y[i]
    stars[id]['e']     = eccentricity[i]
    stars[id]['flux'] = kron_flux[i]
    stars[id]['max']   = max_value[i]

# writing data to file
with open (file_output, 'w') as fh_out:
    # writing header
    fh_out.write ("# star ID, x, y, eccentricity, flux, max value, RA, Dec\n")
    # writing data of individual star
    for id in sorted (stars, key=lambda x: int (stars[x]['flux']), \
                      reverse=True):
        # if peak count is more than 35000 ADU, then skip
        if (stars[id]['max'] > 35000.0):
            continue
        # if the source is like to be a cosmic ray, then skip
        if (stars[id]['max'] / stars[id]['flux'] > 0.1):
            continue
        # if the source is highly elongated, then skip
        if (stars[id]['e'] > 0.5):
            continue
        # if the source is near the edge of the image, then skip
        if ( (stars[id]['x'] < 100.0) \
             or (stars[id]['x'] > header['NAXIS1'] - 100.0) \
             or (stars[id]['y'] < 100.0) \
             or (stars[id]['y'] > header['NAXIS2'] - 100.0) ):
            continue
        # conversion from pixel coordinate into sky coordinate
```

```
        coord = wcs.pixel_to_world (stars[id]['x'], stars[id]['y'])
        # printing information of a source
        fh_out.write ("%02d %8.3f %8.3f %5.3f %10.2f %8.2f %s\n" \
                      % (int (id), stars[id]['x'], stars[id]['y'], \
                         stars[id]['e'], stars[id]['flux'], \
                         stars[id]['max'], coord.to_string ('hmsdms') ) )
```

Execute the script.

```
% chmod a+x ao2021_s21_06.py
% ./ao2021_s21_06.py -h
usage: ao2021_s21_06.py [-h] [-f FILE_FITS] [-t FILE_VOTABLE] [-o FILE_OUTPUT]

Reading a VOTable file and printing information of extracted sources

optional arguments:
  -h, --help            show this help message and exit
  -f FILE_FITS, --file-fits FILE_FITS
                        input FITS file (*.fits)
  -t FILE_VOTABLE, --file-votable FILE_VOTABLE
                        input VOTable file (*.vot)
  -o FILE_OUTPUT, --file-output FILE_OUTPUT
                        output file

% ./ao2021_s21_06.py -f v0678vir/lot_20210214_0117_df.fits -t 0117.vot \
? -o 0117_sources.data
WARNING: FITSFixedWarning: RADECSYS= 'FK5 ' / Equatorial coordinate system
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
% ls -l 0117_sources.data
-rw-r--r--  1 daisuke  taiwan  1068 Jun  3 23:22 0117_sources.data
% cat 0117_sources.data
# star ID, x, y, eccentricity, flux, max value, RA, Dec
22  495.585 1830.242 0.224  374434.34 13744.63 12h32m03.0136s -02d10m26.1321s
19  182.651 1531.658 0.219  327013.01 12097.86 12h32m10.9183s -02d08m29.1056s
23 1889.188 1856.780 0.479  259047.20  8890.20 12h31m27.2261s -02d10m45.6937s
12 1063.511 1134.296 0.222  235968.11  7748.14 12h31m48.1131s -02d06m02.0894s
02  133.225  148.567 0.106  197261.49  6582.33 12h32m11.5668s -01d59m36.4443s
04  691.686  240.190 0.209  149800.83  4509.61 12h31m57.2635s -02d00m15.4649s
13 1397.649 1270.221 0.345  134787.09  3839.70 12h31m39.5908s -02d06m56.6455s
17 1320.865 1341.494 0.359  110961.66  3070.22 12h31m41.595s -02d07m23.5627s
07  749.824  492.788 0.337  103399.12  3003.35 12h31m55.8834s -02d01m53.0764s
18  807.064 1399.006 0.368   94481.64  2795.44 12h31m54.8191s -02d07m42.2501s
14  792.364 1271.415 0.455   64863.17  1880.18 12h31m55.1395s -02d06m53.0436s
05  387.509  406.566 0.192   53821.08  1506.76 12h32m05.1511s -02d01m17.455s
03  497.006  216.317 0.426   43072.01  1368.99 12h32m02.2533s -02d00m04.9678s
```

Make a Python script to show locations of selected stars.

Python Code 7: ao2021_s21_07.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing sys module
import sys
```

```
# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits
import astropy.visualization
import astropy.wcs

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# constructing parser object
desc = "Reading a file and plotting locations of stars"
parser = argparse.ArgumentParser (description=desc)

# list of colours
list_colours = ['maroon', 'red', 'coral', 'bisque', 'orange', \
                'wheat', 'yellow', 'green', 'lime', 'aqua', \
                'skyblue', 'blue', 'indigo', 'violet', 'pink']

# list of markers
list_markers = ['o', 'v', '^', 's', 'p', 'h', '8']

# list of cmap
list_cmap = [ 'viridis', 'plasma', 'inferno', 'magma', 'cividis', 'gray', \
              'bone', 'pink', 'spring', 'summer', 'autumn', 'winter', \
              'cool', 'hot', 'copper', 'hsv', 'ocean', 'terrain', 'gnuplot', \
              'rainbow', 'turbo'
            ]

# adding arguments
parser.add_argument ('-f', '--file-fits', default='', \
                     help='input FITS file (*.fits)')
parser.add_argument ('-i', '--file-input', default='', \
                     help='input file')
parser.add_argument ('-o', '--file-output', default='', \
                     help='output graphics file (EPS, PDF, PNG, or PS file)')
parser.add_argument ('-b', '--cmap', choices=list_cmap, default='bone', \
                     help='cmap for colour bar (default: bone)')
parser.add_argument ('-c', '--colour', choices=list_colours, default='red', \
                     help='colour of marker (default: red)')
parser.add_argument ('-m', '--marker', choices=list_markers, default='o', \
                     help='shape of marker (default: o)')
parser.add_argument ('-r', '--radius', type=float, default=10.0, \
                     help='radius of marker in pixel (default: 10)')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_fits    = args.file_fits
file_input   = args.file_input
file_output  = args.file_output
cmap         = args.cmap
colour       = args.colour
marker       = args.marker
radius       = args.radius
```

```
# check of input file
if not (file_fits[-5:] == '.fits'):
    print ("Input file given by -f option must be a FITS file.")
    sys.exit ()
if (file_input == ''):
    print ("Input file name must be specified.")
    sys.exit ()

# check of output file
if not ( (file_output[-4:] == '.eps') or (file_output[-4:] == '.pdf') \
         or (file_output[-4:] == '.png') or (file_output[-3:] == '.ps') ):
    print ("Output file must be either EPS, PDF, PNG, or PS file.")
    sys.exit ()

# function to read header and image of a FITS file
def read_fits (file_fits):
    # opening FITS file
    with astropy.io.fits.open (file_fits) as hdu:
        # reading header and image
        header = hdu[0].header
        image  = hdu[0].data
        # if no image in PrimaryHDU, then read next HDU
        if (header['NAXIS'] == 0):
            header = hdu[1].header
            image  = hdu[1].data
    # returning header and image
    return (header, image)

# making empty lists for data
list_x = []
list_y = []

# opening input file
with open (file_input, 'r') as fh_in:
    # reading file
    for line in fh_in:
        # skip, if line starts with '#'
        if (line[0] == '#'):
            continue
        # splitting data
        record = line.split ()
        # x and y coordinates
        x = float (record[1])
        y = float (record[2])
        # appending data to lists
        list_x.append (x)
        list_y.append (y)

# reading FITS file
header, image = read_fits (file_fits)
wcs           = astropy.wcs.WCS (header)

# making objects "fig" and "ax"
fig = matplotlib.figure.Figure ()
matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax = fig.add_subplot (111)

# plotting image
norm \
```

```
    = astropy.visualization.mpl_normalize.ImageNormalize \
    ( stretch=astropy.visualization.HistEqStretch (image) )
im = ax.imshow (image, cmap=cmap, norm=norm, origin='upper')
ax.plot (list_x, list_y, marker=marker, color=colour, \
         markersize=radius, fillstyle='none', linestyle='None')
ax.set_xlabel ('X [pixel]')
ax.set_ylabel ('Y [pixel]')

# saving to file
fig.savefig (file_output, dpi=225)
```

Run the script, and make a PDF file.

```
% chmod a+x ao2021_s21_07.py
% ./ao2021_s21_07.py -h
usage: ao2021_s21_07.py [-h] [-f FILE_FITS] [-i FILE_INPUT] [-o FILE_OUTPUT]
                        [-b {viridis,plasma,inferno,magma,cividis,gray,bone,pink
,spring,summer,autumn,winter,cool,hot,copper,hsv,ocean,terrain,gnuplot,rainbow,t
urbo}]
                        [-c {maroon,red,coral,bisque,orange,wheat,yellow,green,l
ime,aqua,skyblue,blue,indigo,violet,pink}]
                        [-m {o,v,^,s,p,h,8}] [-r RADIUS]

Reading a file and plotting locations of stars

optional arguments:
  -h, --help            show this help message and exit
  -f FILE_FITS, --file-fits FILE_FITS
                        input FITS file (*.fits)
  -i FILE_INPUT, --file-input FILE_INPUT
                        input file
  -o FILE_OUTPUT, --file-output FILE_OUTPUT
                        output graphics file (EPS, PDF, PNG, or PS file)
  -b {viridis,plasma,inferno,magma,cividis,gray,bone,pink,spring,summer,autumn,w
inter,cool,hot,copper,hsv,ocean,terrain,gnuplot,rainbow,turbo}, --cmap {viridis,
plasma,inferno,magma,cividis,gray,bone,pink,spring,summer,autumn,winter,cool,hot
,copper,hsv,ocean,terrain,gnuplot,rainbow,turbo}
                        cmap for colour bar (default: bone)
  -c {maroon,red,coral,bisque,orange,wheat,yellow,green,lime,aqua,skyblue,blue,i
ndigo,violet,pink}, --colour {maroon,red,coral,bisque,orange,wheat,yellow,green,
lime,aqua,skyblue,blue,indigo,violet,pink}
                        colour of marker (default: red)
  -m {o,v,^,s,p,h,8}, --marker {o,v,^,s,p,h,8}
                        shape of marker (default: o)
  -r RADIUS, --radius RADIUS
                        radius of marker in pixel (default: 10)

% ./ao2021_s21_07.py -i 0117_sources.data \
? -f v0678vir/lot_20210214_0117_df.fits -o 0117_selected.pdf
WARNING: FITSFixedWarning: RADECSYS= 'FK5 ' / Equatorial coordinate system
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
% ls -l 0117_selected.pdf
-rw-r--r--  1 daisuke  taiwan  1854519 Jun  3 23:38 0117_selected.pdf
```

Use the command `xpdf` to show the PDF file. (Fig. 5)

```
% xpdf 0117_selected.pdf
```

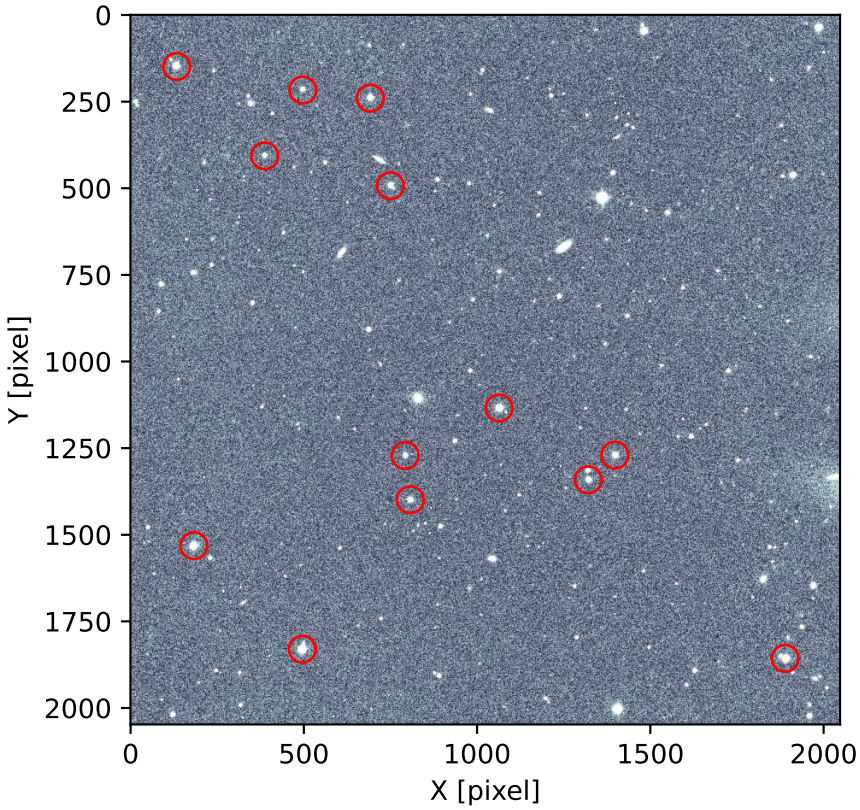

Figure 5: Locations of selected field stars (and V0678 Vir).

9 Aperture photometry

Make a Python script to carry out aperture photometry for V0678 Vir and selected field stars.

Python Code 8: ao2021_s21_08.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing sys module
import sys

# importing datetime module
import datetime

# importing numpy module
import numpy
import numpy.ma

# importing astropy module
import astropy.io.fits
import astropy.modeling
import astropy.units

# importing photutils module
import photutils.aperture

# units
u_ha  = astropy.units.hourangle
u_deg = astropy.units.deg

# choice of PSF model
list_psf_model = ['2dg', '2dm']

# constructing parser object
desc = "Aperture photometry"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-t', '--file-target', default='', \
                     help='target object file')
parser.add_argument ('-o', '--file-output', default='', \
                     help='output file')
parser.add_argument ('-w', '--width', type=int, default=10, \
                     help='half-width of PSF fitting box (default: 10)')
parser.add_argument ('-p', '--psf-model', choices=list_psf_model, \
                     default='2dg', \
                     help='PSF model [2dg=Gaussian, 2dm=Moffat] (default: 2dg)')
parser.add_argument ('-r', '--radius', type=float, default=1.5, \
                     help='radius of aperture in FWHM (default: 1.5)')
parser.add_argument ('-a', '--sky-annulus-inner', type=float, default=5.0, \
                     help='radius of inner sky annulus in FWHM (default: 4)')
parser.add_argument ('-b', '--sky-annulus-outer', type=float, default=8.0, \
                     help='radius of outer sky annulus in FWHM (default: 6)')
parser.add_argument ('-f', '--fwhm-min', type=float, default=2.0, \
                     help='minimum acceptable FWHM in pixel (default: 2)')
parser.add_argument ('-g', '--fwhm-max', type=float, default=8.0, \
                     help='maximum acceptable FWHM in pixel (default: 8)')
```

```
parser.add_argument ('files', nargs='+', help='FITS files')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_target            = args.file_target
file_output            = args.file_output
files_fits             = args.files
half_width             = args.width
psf_model              = args.psf_model
aperture_radius_fwhm   = args.radius
sky_annulus_inner_fwhm = args.sky_annulus_inner
sky_annulus_outer_fwhm = args.sky_annulus_outer
fwhm_min               = args.fwhm_min
fwhm_max               = args.fwhm_max

# check of match file
if (file_target == ''):
    print ("Target object file must be specified.")
    sys.exit ()

# check of output file
if (file_output == ''):
    print ("Output file must be specified.")
    sys.exit ()

# function to read a FITS file
def read_fits (file_fits):
    # reading FITS file
    with astropy.io.fits.open (file_fits) as hdu:
        # reading header and image
        header = hdu[0].header
        image  = hdu[0].data
        # if no image in PrimaryHDU, then read next HDU
        if (header['NAXIS'] == 0):
            header = hdu[1].header
            image  = hdu[1].data
    # returning header and image
    return (header, image)

# making empty lists for coordinates of stars
list_starid = []
list_ra     = []
list_dec    = []

# opening file
with open (file_target, 'r') as fh_in:
    # reading file file line-by-line
    for line in fh_in:
        # if the line starts with '#', then skip
        if (line[0] == '#'):
            continue
        # splitting data
        (star_id, x_str, y_str, eccentricity, flux, max_value, ra, dec) \
            = line.split ()
        # appending data to lists
        list_starid.append (star_id)
        list_ra.append (ra)
```

```
        list_dec.append (dec)

# opening file for writing
fh_out = open (file_output, 'w')

# writing header to file
fh_out.write ("# date/time at middle of exposure, star ID, flux, flux error\n")

# processing FITS file one-by-one
for file_fits in files_fits:
    # reading FITS files
    (header, image) = read_fits (file_fits)
    # WCS
    wcs = astropy.wcs.WCS (header)

    # check of WCS information
    if ( (wcs.pixel_scale_matrix[0][0] == 1) \
         and (wcs.pixel_scale_matrix[0][1] == 0) \
         and (wcs.pixel_scale_matrix[1][0] == 0) \
         and (wcs.pixel_scale_matrix[1][1] == 1) \
        ):
        continue

    # date/time of middle of exposure
    datetime_start  = datetime.datetime.fromisoformat (header['DATE-OBS'])
    half_exptime    = datetime.timedelta (seconds=header['EXPTIME'] / 2.0)
    datetime_middle = datetime_start + half_exptime

    # writing data to file
    fh_out.write ("%s" % datetime_middle.isoformat ())

    # aperture photometry
    for i in range ( len (list_ra) ):
        # X and Y coordinates
        coord = astropy.coordinates.SkyCoord (list_ra[i], list_dec[i], \
                                              unit=(u_ha, u_deg) )
        (x, y) = wcs.world_to_pixel (coord)

        if not ( (x - half_width > 0.0) \
                 and (y - half_width > 0.0) \
                 and (x + half_width < header['NAXIS1']) \
                 and (y + half_width < header['NAXIS1']) ):
            continue

        # region of calculation
        box_xmin = int (x) - half_width
        box_xmax = int (x) + half_width
        box_ymin = int (y) - half_width
        box_ymax = int (y) + half_width
        box = image[box_ymin:box_ymax, box_xmin:box_xmax]

        # rough background subtraction
        box -= numpy.median (box)

        # PSF fitting
        box_y, box_x = numpy.indices (box.shape)
        if (psf_model == '2dg'):
            psf_init = astropy.modeling.models.Gaussian2D (x_mean=half_width, \
                                                           y_mean=half_width)
```

```
        elif (psf_model == '2dm'):
            psf_init = astropy.modeling.models.Moffat2D (x_0=half_width, \
                                                         y_0=half_width)
        fit = astropy.modeling.fitting.LevMarLSQFitter ()
        psf_fitted = fit (psf_init, box_x, box_y, box, maxiter=1000)

        # results of fitting
        if (psf_model == '2dg'):
            x_centre = psf_fitted.x_mean.value
            y_centre = psf_fitted.y_mean.value
            fwhm_x   = psf_fitted.x_fwhm
            fwhm_y   = psf_fitted.y_fwhm
            fwhm     = (fwhm_x + fwhm_y) / 2.0
        elif (psf_model == '2dm'):
            x_centre = psf_fitted.x_0.value
            y_centre = psf_fitted.y_0.value
            fwhm     = psf_fitted.fwhm

        # check of measured FWHM
        if not ( (fwhm > fwhm_min) and (fwhm < fwhm_max) ):
            continue

        # aperture radius in pixel
        aperture_radius_pix = fwhm * aperture_radius_fwhm

        # sky annulus
        sky_annulus_inner_pix = fwhm * sky_annulus_inner_fwhm
        sky_annulus_outer_pix = fwhm * sky_annulus_outer_fwhm

        # aperture
        aperture \
            = photutils.aperture.CircularAperture ((x, y), \
                                                   r=aperture_radius_pix)

        # sky annulus
        annulus \
            = photutils.aperture.CircularAnnulus ((x, y), \
                                                  r_in=sky_annulus_inner_pix, \
                                                  r_out=sky_annulus_outer_pix)

        # masked data for sky annulus
        skyannulus_data  = annulus.to_mask (method='center').multiply (image)
        skyannulus_mask  = skyannulus_data <= 0.0
        skyannulus_mdata = numpy.ma.array (skyannulus_data, \
                                           mask=skyannulus_mask)

        # sky background estimate using sigma-clipping algorithm
        skybg_mean, skybg_median, skybg_stddev \
            = astropy.stats.sigma_clipped_stats (skyannulus_mdata, \
                                                 sigma=3.0, maxiters=10, \
                                                 cenfunc='median')

        # sky background per pixel
        skybg_per_pixel = 3.0 * skybg_median - 2.0 * skybg_mean

        # aperture photometry
        image_mask   = image < 0.0
        image_masked = numpy.ma.array (image, mask=image_mask)
        noise        = numpy.sqrt (image_masked)
```

```
        phot_star    = photutils.aperture.aperture_photometry (image, \
                                                               aperture, \
                                                               error=noise)

        # net flux
        net_flux = phot_star['aperture_sum'] - skybg_per_pixel * aperture.area

        # error
        flux_err = phot_star['aperture_sum_err']

        # rejecting some data
        if (net_flux < 0.0):
            continue

        # writing data to file
        fh_out.write (" %s,%.4f,%.4f,%s,%s" \
                      % (list_starid[i], net_flux, flux_err, \
                         list_ra[i], list_dec[i]) )

    # writing data to file
    fh_out.write ("\n")

# closing file
fh_out.close ()
```

Run the script and carry out aperture photometry.

```
% chmod a+x ao2021_s21_08.py
% ./ao2021_s21_08.py -h
usage: ao2021_s21_08.py [-h] [-t FILE_TARGET] [-o FILE_OUTPUT] [-w WIDTH]
                        [-p {2dg,2dm}] [-r RADIUS] [-a SKY_ANNULUS_INNER]
                        [-b SKY_ANNULUS_OUTER] [-f FWHM_MIN] [-g FWHM_MAX]
                        files [files ...]

Aperture photometry

positional arguments:
  files                 FITS files

optional arguments:
  -h, --help            show this help message and exit
  -t FILE_TARGET, --file-target FILE_TARGET
                        target object file
  -o FILE_OUTPUT, --file-output FILE_OUTPUT
                        output file
  -w WIDTH, --width WIDTH
                        half-width of PSF fitting box (default: 10)
  -p {2dg,2dm}, --psf-model {2dg,2dm}
                        PSF model [2dg=Gaussian, 2dm=Moffat] (default: 2dg)
  -r RADIUS, --radius RADIUS
                        radius of aperture in FWHM (default: 1.5)
  -a SKY_ANNULUS_INNER, --sky-annulus-inner SKY_ANNULUS_INNER
                        radius of inner sky annulus in FWHM (default: 4)
  -b SKY_ANNULUS_OUTER, --sky-annulus-outer SKY_ANNULUS_OUTER
                        radius of outer sky annulus in FWHM (default: 6)
  -f FWHM_MIN, --fwhm-min FWHM_MIN
                        minimum acceptable FWHM in pixel (default: 2)
  -g FWHM_MAX, --fwhm-max FWHM_MAX
```

```
                        maximum acceptable FWHM in pixel (default: 8)

% ./ao2021_s21_08.py -t 0117_sources.data -o phot.data v0678vir/*.fits
% ls -l phot.data
-rw-r--r--  1 daisuke  taiwan  10928 Jun  4 01:28 phot.data
```