Advanced Astronomical Observations 2021
Session 15: Atmospheric Extinction

Kinoshita Daisuke

12 May 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 study atmospheric extinction.

1 Copying the data

We process the data of the standard star field PG 1047+003. Check the data taken on 14 February 2021, and search for reduced FITS files of the standard star field PG 1047+003.

Python Code 1: ao2021_s15_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 = "checking FITS header"
parser = argparse.ArgumentParser (description=desc)
```

```
# adding argument
parser.add_argument ('-d', '--dir', default='', help='name of data directory')
parser.add_argument ('-n', '--name', default='', help='name of target object')

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

# input parameters
dir_data    = args.dir
target_name = args.name

# checking "dir_data" and "target_name"
if (dir_data == ''):
    print ("You have to specify data directory name by using -d option!")
    sys.exit ()
if (target_name == ''):
    print ("You have to specify target object name by using -n option!")
    sys.exit ()

# making a pathlib object for "dir_data"
path_datadir = pathlib.Path (dir_data)
# if not a directory, then stop the script
if not (path_datadir.is_dir () ):
    print ("%s is not a directory!" % dir_data)
    sys.exit ()
# reading directory
files_fits = path_datadir.iterdir ()

# printing header
print ("# file name, time-obs, data type, exptime, filter, object, airmass")

# processing each FITS file
for path_fits in files_fits:
    # file name
    file_fits = str (path_fits)
    # if the file is not a FITS file, then skip
    if not (file_fits[-5:] == '.fits'):
        continue
    # if the file is not a reduced data, then skip
    if not (file_fits[-8:] == '_df.fits'):
        continue
    # opening a FITS file
    with astropy.io.fits.open (file_fits) as hdu_list:
        # reading header information
        header = hdu_list[0].header
    # checking header information
    if not ( (header['IMAGETYP'] == 'LIGHT') \
             and (header['OBJECT'] == target_name) ):
        continue
    # printing file information
    print ("%s %s %s %6.1f %s %s %5.3f" \
           % (path_fits.name, header['TIME-OBS'], header['IMAGETYP'], \
              header['EXPTIME'], header['FILTER'], header['OBJECT'], \
              header['AIRMASS']) )
```

Execute the script to find reduced FITS files of the standard star field PG 1047+003.

```
% chmod a+x ao2021_s15_01.py
% ./ao2021_s15_01.py -h
usage: ao2021_s15_01.py [-h] [-d DIR] [-n NAME]

checking FITS header

optional arguments:
  -h, --help            show this help message and exit
  -d DIR, --dir DIR     name of data directory
  -n NAME, --name NAME  name of target object

% ./ao2021_s15_01.py -n PG1047 \
? -d ../../14_reduction3/script_14/ccdred_20210511_122633
# file name, time-obs, data type, exptime, filter, object, airmass
lot_20210214_0145_df.fits 17:24:46 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.094
lot_20210214_0146_df.fits 17:25:13 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.094
lot_20210214_0147_df.fits 17:25:40 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.095
lot_20210214_0148_df.fits 17:26:15 LIGHT   10.0 rp_Astrodon_2019 PG1047 1.095
lot_20210214_0149_df.fits 17:26:41 LIGHT   10.0 rp_Astrodon_2019 PG1047 1.095
lot_20210214_0150_df.fits 17:27:09 LIGHT   10.0 rp_Astrodon_2019 PG1047 1.095
lot_20210214_0151_df.fits 17:27:45 LIGHT   10.0 ip_Astrodon_2019 PG1047 1.095
lot_20210214_0152_df.fits 17:28:12 LIGHT   10.0 ip_Astrodon_2019 PG1047 1.096
lot_20210214_0153_df.fits 17:28:39 LIGHT   10.0 ip_Astrodon_2019 PG1047 1.096
lot_20210214_0185_df.fits 17:58:41 LIGHT  180.0 gp_Astrodon_2019 PG1047 1.122
lot_20210214_0186_df.fits 18:01:59 LIGHT  180.0 gp_Astrodon_2019 PG1047 1.126
lot_20210214_0187_df.fits 18:05:25 LIGHT  180.0 rp_Astrodon_2019 PG1047 1.131
lot_20210214_0188_df.fits 18:08:41 LIGHT  180.0 rp_Astrodon_2019 PG1047 1.135
lot_20210214_0189_df.fits 18:12:06 LIGHT  180.0 ip_Astrodon_2019 PG1047 1.140
lot_20210214_0190_df.fits 18:15:23 LIGHT  180.0 ip_Astrodon_2019 PG1047 1.145
lot_20210214_0245_df.fits 19:12:43 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.277
lot_20210214_0246_df.fits 19:13:10 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.279
lot_20210214_0247_df.fits 19:13:36 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.280
lot_20210214_0248_df.fits 19:14:11 LIGHT   10.0 rp_Astrodon_2019 PG1047 1.282
lot_20210214_0249_df.fits 19:14:38 LIGHT   10.0 rp_Astrodon_2019 PG1047 1.284
lot_20210214_0250_df.fits 19:15:04 LIGHT   10.0 rp_Astrodon_2019 PG1047 1.285
lot_20210214_0251_df.fits 19:15:40 LIGHT   10.0 ip_Astrodon_2019 PG1047 1.288
lot_20210214_0252_df.fits 19:16:07 LIGHT   10.0 ip_Astrodon_2019 PG1047 1.289
lot_20210214_0253_df.fits 19:16:33 LIGHT   10.0 ip_Astrodon_2019 PG1047 1.291
lot_20210214_0285_df.fits 19:42:32 LIGHT   60.0 gp_Astrodon_2019 PG1047 1.404
lot_20210214_0286_df.fits 19:43:48 LIGHT   60.0 gp_Astrodon_2019 PG1047 1.410
lot_20210214_0287_df.fits 19:45:04 LIGHT   60.0 gp_Astrodon_2019 PG1047 1.417
lot_20210214_0288_df.fits 19:46:29 LIGHT   60.0 rp_Astrodon_2019 PG1047 1.424
lot_20210214_0289_df.fits 19:47:46 LIGHT   60.0 rp_Astrodon_2019 PG1047 1.431
lot_20210214_0290_df.fits 19:49:04 LIGHT   60.0 rp_Astrodon_2019 PG1047 1.438
lot_20210214_0291_df.fits 19:50:30 LIGHT   60.0 ip_Astrodon_2019 PG1047 1.446
lot_20210214_0292_df.fits 19:51:46 LIGHT   60.0 ip_Astrodon_2019 PG1047 1.453
lot_20210214_0293_df.fits 19:53:03 LIGHT   60.0 ip_Astrodon_2019 PG1047 1.460
lot_20210214_0325_df.fits 20:18:59 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.628
lot_20210214_0326_df.fits 20:19:25 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.631
lot_20210214_0327_df.fits 20:19:51 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.635
lot_20210214_0328_df.fits 20:20:27 LIGHT   10.0 rp_Astrodon_2019 PG1047 1.640
lot_20210214_0329_df.fits 20:20:54 LIGHT   10.0 rp_Astrodon_2019 PG1047 1.643
lot_20210214_0330_df.fits 20:21:19 LIGHT   10.0 rp_Astrodon_2019 PG1047 1.647
lot_20210214_0331_df.fits 20:21:55 LIGHT   10.0 ip_Astrodon_2019 PG1047 1.652
lot_20210214_0332_df.fits 20:22:21 LIGHT   10.0 ip_Astrodon_2019 PG1047 1.655
lot_20210214_0333_df.fits 20:22:48 LIGHT   10.0 ip_Astrodon_2019 PG1047 1.659
lot_20210214_0366_df.fits 21:00:14 LIGHT   10.0 gp_Astrodon_2019 PG1047 2.072
lot_20210214_0367_df.fits 21:00:41 LIGHT   10.0 gp_Astrodon_2019 PG1047 2.079
lot_20210214_0368_df.fits 21:01:07 LIGHT   10.0 gp_Astrodon_2019 PG1047 2.086
```

```
lot_20210214_0369_df.fits 21:01:43 LIGHT   10.0 rp_Astrodon_2019 PG1047 2.094
lot_20210214_0370_df.fits 21:02:09 LIGHT   10.0 rp_Astrodon_2019 PG1047 2.101
lot_20210214_0371_df.fits 21:02:36 LIGHT   10.0 rp_Astrodon_2019 PG1047 2.108
lot_20210214_0372_df.fits 21:03:12 LIGHT   10.0 ip_Astrodon_2019 PG1047 2.117
lot_20210214_0373_df.fits 21:03:38 LIGHT   10.0 ip_Astrodon_2019 PG1047 2.124
lot_20210214_0374_df.fits 21:04:05 LIGHT   10.0 ip_Astrodon_2019 PG1047 2.130
% ./ao2021_s15_01.py -d ../../14_reduction3/script_14/ccdred_20210511_122633 \
? -n PG1047 | grep -v # | wc
      51     357    3978
```

51 files are found. Next, we copy these 51 files.

Python Code 2: ao2021_s15_02.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing shutil module
import shutil

# importing astropy module
import astropy.io.fits

# constructing parser object
desc = "copying FITS files"
parser = argparse.ArgumentParser (description=desc)

# adding argument
parser.add_argument ('-d', '--dir', default='', help='name of data directory')
parser.add_argument ('-n', '--name', default='', help='name of target object')

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

# input parameters
dir_data    = args.dir
target_name = args.name

# checking "dir_data" and "target_name"
if (dir_data == ''):
    print ("You have to specify data directory name by using -d option!")
    sys.exit ()
if (target_name == ''):
    print ("You have to specify target object name by using -n option!")
    sys.exit ()

# making a pathlib object for "dir_data"
path_datadir = pathlib.Path (dir_data)
# if not a directory, then stop the script
if not (path_datadir.is_dir () ):
    print ("%s is not a directory!" % dir_data)
```

```
    sys.exit ()
# reading directory
files_fits = path_datadir.iterdir ()

# processing each FITS file
for path_fits in files_fits:
    # file name
    file_fits = str (path_fits)
    # if the file is not a FITS file, then skip
    if not (file_fits[-5:] == '.fits'):
        continue
    # if the file is not a reduced data, then skip
    if not (file_fits[-8:] == '_df.fits'):
        continue
    # opening a FITS file
    with astropy.io.fits.open (file_fits) as hdu_list:
        # reading header information
        header = hdu_list[0].header
    # checking header information
    if not ( (header['IMAGETYP'] == 'LIGHT') \
             and (header['OBJECT'] == target_name) ):
        continue
    # printing status
    print ("Now copying the file %s..." % path_fits.name)
    # copying the file
    shutil.copy2 (file_fits, '.')
    # printing status
    print ("Finished copying the file %s!" % path_fits.name)
```

Run the script, and copy files.

```
% chmod a+x ao2021_s15_02.py
% ./ao2021_s15_02.py -h
usage: ao2021_s15_02.py [-h] [-d DIR] [-n NAME]

copying FITS files

optional arguments:
  -h, --help            show this help message and exit
  -d DIR, --dir DIR     name of data directory
  -n NAME, --name NAME  name of target object

% ls
ao2021_s15_01.py*   ao2021_s15_01.py~   ao2021_s15_02.py*   ao2021_s15_02.py~
% mkdir pg1047
% ls
ao2021_s15_01.py*   ao2021_s15_02.py*   pg1047/
ao2021_s15_01.py~   ao2021_s15_02.py~
% cd pg1047
% ls
% ../ao2021_s15_02.py -n PG1047 \
? -d ../../../14_reduction3/script_14/ccdred_20210511_122633
Now copying the file lot_20210214_0145_df.fits...
Finished copying the file lot_20210214_0145_df.fits!
Now copying the file lot_20210214_0146_df.fits...
Finished copying the file lot_20210214_0146_df.fits!
Now copying the file lot_20210214_0147_df.fits...
Finished copying the file lot_20210214_0147_df.fits!
```

```
Now copying the file lot_20210214_0148_df.fits...
Finished copying the file lot_20210214_0148_df.fits!
Now copying the file lot_20210214_0149_df.fits...
Finished copying the file lot_20210214_0149_df.fits!

.....

Now copying the file lot_20210214_0370_df.fits...
Finished copying the file lot_20210214_0370_df.fits!
Now copying the file lot_20210214_0371_df.fits...
Finished copying the file lot_20210214_0371_df.fits!
Now copying the file lot_20210214_0372_df.fits...
Finished copying the file lot_20210214_0372_df.fits!
Now copying the file lot_20210214_0373_df.fits...
Finished copying the file lot_20210214_0373_df.fits!
Now copying the file lot_20210214_0374_df.fits...
Finished copying the file lot_20210214_0374_df.fits!
% ls
lot_20210214_0145_df.fits               lot_20210214_0287_df.fits
lot_20210214_0146_df.fits               lot_20210214_0288_df.fits
lot_20210214_0147_df.fits               lot_20210214_0289_df.fits
lot_20210214_0148_df.fits               lot_20210214_0290_df.fits
lot_20210214_0149_df.fits               lot_20210214_0291_df.fits

.....

lot_20210214_0251_df.fits               lot_20210214_0371_df.fits
lot_20210214_0252_df.fits               lot_20210214_0372_df.fits
lot_20210214_0253_df.fits               lot_20210214_0373_df.fits
lot_20210214_0285_df.fits               lot_20210214_0374_df.fits
lot_20210214_0286_df.fits
% cd ..
% ls
ao2021_s15_01.py*   ao2021_s15_02.py*   pg1047/
ao2021_s15_01.py~   ao2021_s15_02.py~
% du -m
1636    ./pg1047
1636    .
```

2 Standard star catalogue

2.1 Downloading standard star catalogue

Download the standard star catalogue for SDSS photometric system.

```
% curl -k -o stds.tar.gz \
? https://www-star.fnal.gov/NorthEqExtension_ugriz/Data/usno40stds.clean.v3.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 65774  100 65774    0     0  17971      0  0:00:03  0:00:03 --:--:-- 17971
% mkdir stds
% cd stds
% tar xzvf stds.tar.gz
x 100_a.txt.clean.v3
x 100_b.txt.clean.v3
x 101_a.txt.clean.v3
x 101_c.txt.clean.v3
x 104_a.txt.clean.v3
```

```
.....

x Ross838.txt.clean.v3
x Ru149.txt.clean.v3
x Wolf1346.txt.clean.v3
x Wolf1447.txt.clean.v3
x Wolf365.txt.clean.v3
% ls
100_a.txt.clean.v3        97_a.txt.clean.v3         GCRV5951.txt.clean.v3
100_b.txt.clean.v3        97_b.txt.clean.v3         GCRV7017.txt.clean.v3
101_a.txt.clean.v3        97_c.txt.clean.v3         GCRV7951.txt.clean.v3
101_c.txt.clean.v3        97_d.txt.clean.v3         GCRV8758.txt.clean.v3
104_a.txt.clean.v3        98_a.txt.clean.v3         GCRV9438.txt.clean.v3

.....

95_b.txt.clean.v3         G15-24.txt.clean.v3       Ru149.txt.clean.v3
95_f.txt.clean.v3         G163_50-51.txt.clean.v3   Wolf1346.txt.clean.v3
96_a.txt.clean.v3         G27-45.txt.clean.v3       Wolf1447.txt.clean.v3
96_b.txt.clean.v3         G3-33.txt.clean.v3        Wolf365.txt.clean.v3
96_c.txt.clean.v3         GCRV5757.txt.clean.v3
% cd ..
```

2.2 Coordinates of photometric standards in PG 1047+003 field

Show the coordinates of photometric standard stars in the field PG 1047+003.

Python Code 3: ao2021_s15_03.py

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

# importing astropy module
import astropy.coordinates

# data file name
file_data = 'stds/PG1047+003A.txt.clean.v3'

# printing header
print ("# ID, RA (deg), Dec (deg), RA (hex), Dec (hex), g' mag, r' mag, i' mag")

# opening the file
with open (file_data, 'r') as fh:
    # reading the file line-by-line
    for line in fh:
        # if the line starts with '#', then skip
        if (line[0] == '#'):
            continue
        # splitting the data
        records = line.split ()
        # star ID
        star_id = int (records[0])
        # RA
        ra_deg  = float (records[1])
        # Dec
        dec_deg = float (records[2])
        # g'-band magnitude
        mag_g   = float (records[6])
        # r'-band magnitude
```

```
        mag_r    = float (records[9])
        # i'-band magnitude
        mag_i    = float (records[12])
        # coordinates
        coord = astropy.coordinates.SkyCoord (ra_deg, dec_deg, unit='deg')
        # conversion into hmsdms format
        radec_str = coord.to_string ('hmsdms')
        (ra_str, dec_str) = radec_str.split ()
        # printing coordinate
        print ("%03d %9.5f %+8.4f %-16s %-16s %6.3f %6.3f %6.3f" \
               % (star_id, ra_deg, dec_deg, ra_str, dec_str, \
                  mag_g, mag_r, mag_i) )
```

Execute the script.

```
% chmod a+x ao2021_s15_03.py
% ./ao2021_s15_03.py > pg1047.list
% cat pg1047.list
# ID, RA (deg), Dec (deg), RA (hex), Dec (hex), g' mag, r' mag, i' mag
004 162.62371  +0.0725 10h50m29.6904s   +00d04m21s        12.127 11.709 11.587
005 162.55708  -0.0090 10h50m13.6992s   -00d00m32.4s      12.826 12.373 12.217
006 162.52363  -0.0198 10h50m05.6712s   -00d01m11.28s     13.819 13.304 13.092
007 162.51184  -0.0102 10h50m02.8416s   -00d00m36.72s     13.240 13.718 14.087
008 162.59599  -0.0818 10h50m23.0376s   -00d04m54.48s     14.511 13.803 13.579
010 162.47199  -0.0596 10h49m53.2776s   -00d03m34.56s     14.797 14.308 14.139
011 162.47082  -0.0579 10h49m52.9968s   -00d03m28.44s     14.814 14.318 14.143
013 162.53305  -0.0346 10h50m07.932s    -00d02m04.56s     15.061 14.553 14.409
014 162.53285  -0.0932 10h50m07.884s    -00d05m35.52s     15.157 14.616 14.440
017 162.57442  -0.0208 10h50m17.8608s   -00d01m14.88s     15.222 14.838 14.707
020 162.61912  +0.0295 10h50m28.5888s   +00d01m46.2s      14.961 14.897 14.946
022 162.55930  -0.0909 10h50m14.232s    -00d05m27.24s     15.415 14.933 14.772
025 162.49926  -0.0433 10h49m59.8224s   -00d02m35.88s     15.861 15.322 15.123
```

3 Processing g'-band data

We first process g'-band data.

3.1 Listing g'-band data

Check the data and list g'-band data.

Python Code 4: ao2021_s15_04.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 = "checking FITS header"
parser = argparse.ArgumentParser (description=desc)

# adding argument
parser.add_argument ('-d', '--dir', default='', help='name of data directory')
parser.add_argument ('-f', '--filter', default='', help='filter name')
parser.add_argument ('-n', '--name', default='', help='name of target object')

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

# input parameters
dir_data    = args.dir
target_name = args.name
filter_name = args.filter

# checking "dir_data", "target_name", and "filter_name"
if (dir_data == ''):
    print ("You have to specify data directory name by using -d option!")
    sys.exit ()
if (target_name == ''):
    print ("You have to specify target object name by using -n option!")
    sys.exit ()
if (filter_name == ''):
    print ("You have to specify filter name by using -f option!")
    sys.exit ()

# making a pathlib object for "dir_data"
path_datadir = pathlib.Path (dir_data)
# if not a directory, then stop the script
if not (path_datadir.is_dir () ):
    print ("%s is not a directory!" % dir_data)
    sys.exit ()
# reading directory
files_fits = path_datadir.iterdir ()

# printing header
print ("# file name, time-obs, data type, exptime, filter, object, airmass")

# processing each FITS file
for path_fits in files_fits:
    # file name
    file_fits = str (path_fits)
    # if the file is not a FITS file, then skip
    if not (file_fits[-5:] == '.fits'):
        continue
    # if the file is not a reduced data, then skip
    if not (file_fits[-8:] == '_df.fits'):
        continue
    # opening a FITS file
    with astropy.io.fits.open (file_fits) as hdu_list:
        # reading header information
        header = hdu_list[0].header
    # checking header information
    if not ( (header['IMAGETYP'] == 'LIGHT') \
             and (header['OBJECT'] == target_name) \
             and (header['FILTER'] == filter_name) ):
        continue
```

```
    # printing file information
    print ("%s %s %s %6.1f %s %s %5.3f" \
           % (path_fits.name, header['TIME-OBS'], header['IMAGETYP'], \
              header['EXPTIME'], header['FILTER'], header['OBJECT'], \
              header['AIRMASS']) )
```

Execute the script to list g'-band data.

```
% chmod a+x ao2021_s15_04.py
% ./ao2021_s15_04.py -h
usage: ao2021_s15_04.py [-h] [-d DIR] [-f FILTER] [-n NAME]

checking FITS header

optional arguments:
  -h, --help            show this help message and exit
  -d DIR, --dir DIR     name of data directory
  -f FILTER, --filter FILTER
                        filter name
  -n NAME, --name NAME  name of target object

% ./ao2021_s15_04.py -n PG1047 -d pg1047 -f gp_Astrodon_2019
# file name, time-obs, data type, exptime, filter, object, airmass
lot_20210214_0145_df.fits 17:24:46 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.094
lot_20210214_0146_df.fits 17:25:13 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.094
lot_20210214_0147_df.fits 17:25:40 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.095
lot_20210214_0185_df.fits 17:58:41 LIGHT  180.0 gp_Astrodon_2019 PG1047 1.122
lot_20210214_0186_df.fits 18:01:59 LIGHT  180.0 gp_Astrodon_2019 PG1047 1.126
lot_20210214_0245_df.fits 19:12:43 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.277
lot_20210214_0246_df.fits 19:13:10 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.279
lot_20210214_0247_df.fits 19:13:36 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.280
lot_20210214_0285_df.fits 19:42:32 LIGHT   60.0 gp_Astrodon_2019 PG1047 1.404
lot_20210214_0286_df.fits 19:43:48 LIGHT   60.0 gp_Astrodon_2019 PG1047 1.410
lot_20210214_0287_df.fits 19:45:04 LIGHT   60.0 gp_Astrodon_2019 PG1047 1.417
lot_20210214_0325_df.fits 20:18:59 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.628
lot_20210214_0326_df.fits 20:19:25 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.631
lot_20210214_0327_df.fits 20:19:51 LIGHT   10.0 gp_Astrodon_2019 PG1047 1.635
lot_20210214_0366_df.fits 21:00:14 LIGHT   10.0 gp_Astrodon_2019 PG1047 2.072
lot_20210214_0367_df.fits 21:00:41 LIGHT   10.0 gp_Astrodon_2019 PG1047 2.079
lot_20210214_0368_df.fits 21:01:07 LIGHT   10.0 gp_Astrodon_2019 PG1047 2.086
```

3.2 Identifying photometric standard star

We carry out aperture photometry for the star ID 8 in the field of PG 1047+003. Make a Python script to mark the location of the star ID 8.

Python Code 5: ao2021_s15_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.coordinates
import astropy.units

# importing photutils module
import photutils.aperture

# importing matplotlib module
import matplotlib.pyplot

# constructing parser object
desc = 'marking target object'
parser = argparse.ArgumentParser (description=desc)

# adding argument
parser.add_argument ('-i', '--input', default='', \
                     help='input FITS file name')
parser.add_argument ('-o', '--output', default='', \
                     help='output image file name')
parser.add_argument ('-r', '--ra', type=float, default=-999.999, \
                     help='RA in degree')
parser.add_argument ('-d', '--dec', type=float, default=-999.999, \
                     help='Dec in degree')
parser.add_argument ('-s', '--size', type=float, default=10.0, \
                     help='radius of aperture in arcsec')
parser.add_argument ('-a', '--z1', type=float, default=0.0, \
                     help='minimum value to display')
parser.add_argument ('-b', '--z2', type=float, default=100.0, \
                     help='maximum value to display')

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

# input parameters
file_fits      = args.input
file_output    = args.output
target_ra_deg  = args.ra
target_dec_deg = args.dec
radius_arcsec  = args.size
z1             = args.z1
z2             = args.z2

# unit
u_arcsec = astropy.units.arcsec

# check of RA and Dec
if ( (target_ra_deg < 0.0) or (target_ra_deg > 360.0) \
     or (target_dec_deg < -90.0) or (target_dec_deg > 90.0) ):
    print ("Something is wrong with RA or Dec!")
    print ("Check RA and Dec you specify.")
    print ("RA  = %f deg" % target_ra_deg)
    print ("Dec = %f deg" % target_dec_deg)
    sys.exit ()

# check of FITS file name
if not (file_fits[-5:] == '.fits'):
    print ("The file \"%s\" is not a FITS file!" % file_fits)
    print ("Check the file name.")
```

```
    sys.exit ()

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

# coordinate
coord = astropy.coordinates.SkyCoord (target_ra_deg, target_dec_deg, unit='deg')

# opening FITS file
with astropy.io.fits.open (file_fits) as hdu_list:
    # reading header information
    header = hdu_list[0].header
    # WCS information
    wcs = astropy.wcs.WCS (header)
    # reading image data
    data   = hdu_list[0].data

# making an aperture
aperture_sky \
    = photutils.aperture.SkyCircularAperture (coord, r=radius_arcsec * u_arcsec)
aperture_pix = aperture_sky.to_pixel (wcs)

# printing aperture information
print ("aperture_sky:")
print (aperture_sky)
print ("aperture_pix:")
print (aperture_pix)

# making plot
matplotlib.pyplot.imshow (data, origin='upper', vmin=z1, vmax=z2)
aperture_pix.plot (color='red', lw=1.0)
matplotlib.pyplot.colorbar ()
matplotlib.pyplot.savefig (file_output, dpi=225)
```

Run the script, and make an image file. The star ID 8 in the field of PG 1047+003 is located at (x, y) = ($\sim$ 559, $\sim$ 1554). (Fig. 1)

```
% chmod a+x ao2021_s15_05.py
% ./ao2021_s15_05.py -h
usage: ao2021_s15_05.py [-h] [-i INPUT] [-o OUTPUT] [-r RA] [-d DEC] [-s SIZE]
                        [-a Z1] [-b Z2]

marking target object

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input FITS file name
  -o OUTPUT, --output OUTPUT
                        output image file name
  -r RA, --ra RA        RA in degree
  -d DEC, --dec DEC     Dec in degree
  -s SIZE, --size SIZE  radius of aperture in arcsec
  -a Z1, --z1 Z1        minimum value to display
```

```
  -b Z2, --z2 Z2        maximum value to display

% ./ao2021_s15_05.py -r 162.59599 -d -0.0818 -i pg1047/lot_20210214_0145_df.fits \
? -s 20 -a 0 -b 50 -o 0145_pg1047_008.pdf
WARNING: FITSFixedWarning: RADECSYS= 'FK5 ' / Equatorial coordinate system
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
aperture_sky:
Aperture: SkyCircularAperture
positions: <SkyCoord (ICRS): (ra, dec) in deg
    (162.59599, -0.0818)>
r: 20.0 arcsec
aperture_pix:
Aperture: CircularAperture
positions: [ 559.36869152, 1553.97257902]
r: 51.947330946799326
% ls -l 0145_pg1047_008.pdf
-rw-r--r--  1 daisuke  taiwan  1472022 May 11 21:12 0145_pg1047_008.pdf
% xpdf 0145_pg1047_008.pdf
```

3.3 Aperture photometry of star ID 8

Make a Python script to carry out aperture photometry of a star at given RA and Dec.

Python Code 6: ao2021_s15_06.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

# importing astropy module
import astropy.coordinates

# importing photutils module
import photutils.centroids
import photutils.aperture

# constructing parser object
desc = 'aperture photometry of a star at given RA and Dec'
parser = argparse.ArgumentParser (description=desc)

# adding argument
parser.add_argument ('-i', '--input', default='', \
                     help='input FITS file name')
parser.add_argument ('-o', '--output', default='', \
                     help='output file name')
parser.add_argument ('-r', '--ra', type=float, default=-999.999, \
                     help='RA in degree')
parser.add_argument ('-d', '--dec', type=float, default=-999.999, \
```

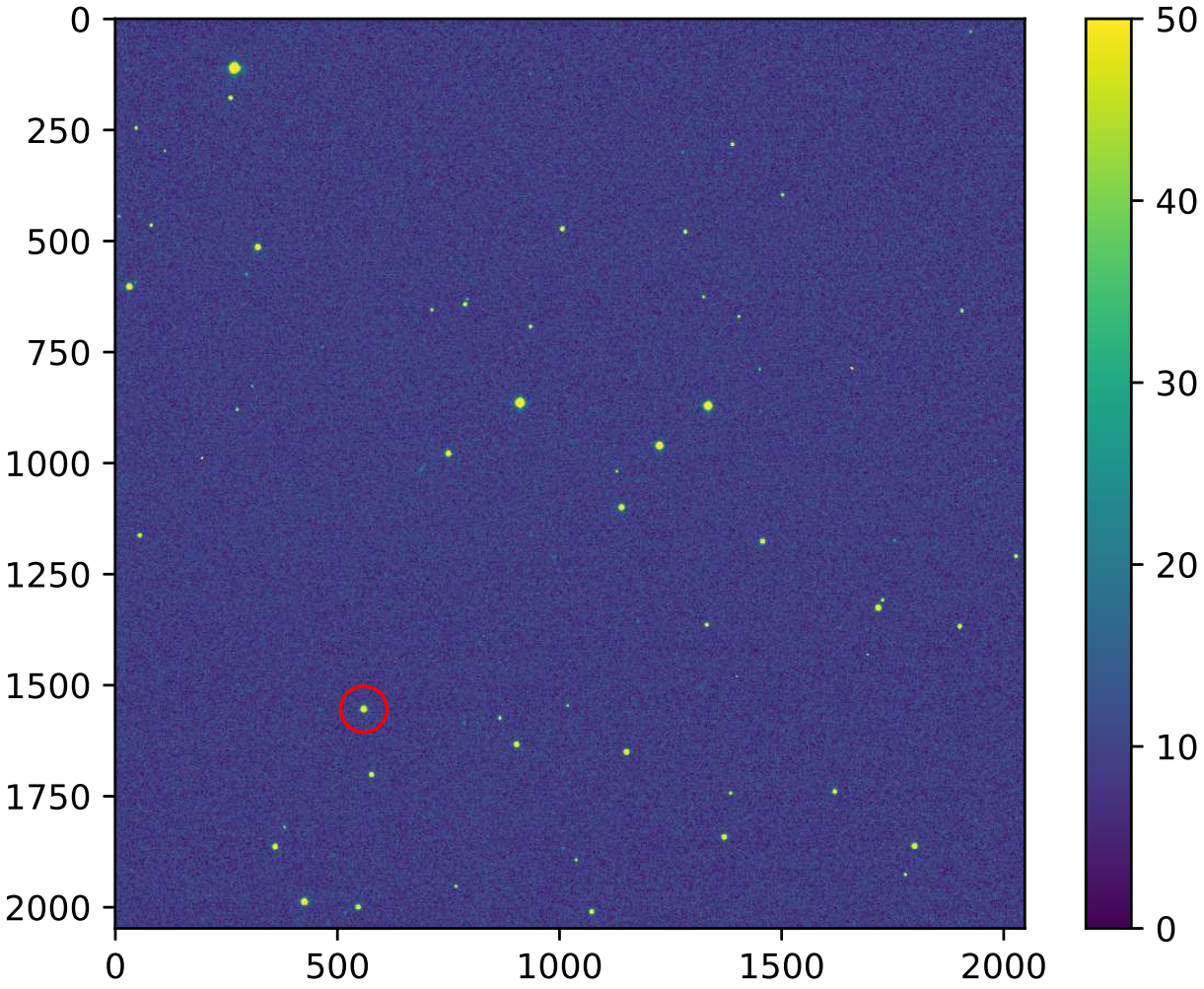

Figure 1: The location of the star ID 8 in the field of PG 1047+003.

```
                      help='Dec in degree')
parser.add_argument ('-a', '--aperture', type=float, default=1.5, \
                      help='aperture radius in FWHM (default: 1.5)')
parser.add_argument ('-w', '--halfwidth', type=int, default=10, \
                      help='half-width for centroid measurement (default: 10)')
parser.add_argument ('-s1', '--skyannulus1', type=float, default=4.0, \
                      help='inner sky annulus radius in FWHM (default: 4)')
parser.add_argument ('-s2', '--skyannulus2', type=float, default=7.0, \
                      help='outer sky annulus radius in FWHM (default: 7)')
parser.add_argument ('-t', '--threshold', type=float, default=3.0, \
                      help='threshold for sigma-clipping in sigma (default: 3)')
parser.add_argument ('-n', '--maxiters', type=int, default=10, \
                      help='maximum number of iterations (default: 10)')
parser.add_argument ('-e', '--keyword-exptime', default='EXPTIME', \
                      help='FITS keyword for exposure time (default: EXPTIME)')
parser.add_argument ('-f', '--keyword-filter', default='FILTER', \
                      help='FITS keyword for filter name (default: FILTER)')
parser.add_argument ('-m', '--keyword-airmass', default='AIRMASS', \
                      help='FITS keyword for airmass (default: AIRMASS)')

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

# input parameters
file_fits             = args.input
file_output           = args.output
target_ra_deg         = args.ra
target_dec_deg        = args.dec
aperture_radius_fwhm  = args.aperture
halfwidth             = args.halfwidth
skyannulus_inner_fwhm = args.skyannulus1
skyannulus_outer_fwhm = args.skyannulus2
threshold             = args.threshold
maxiters              = args.maxiters
keyword_exptime       = args.keyword_exptime
keyword_filter        = args.keyword_filter
keyword_airmass       = args.keyword_airmass

# check of RA and Dec
if ( (target_ra_deg < 0.0) or (target_ra_deg > 360.0) \
     or (target_dec_deg < -90.0) or (target_dec_deg > 90.0) ):
    print ("Something is wrong with RA or Dec!")
    print ("Check RA and Dec you specify.")
    print ("RA  = %f deg" % target_ra_deg)
    print ("Dec = %f deg" % target_dec_deg)
    sys.exit ()

# check of FITS file name
if not (file_fits[-5:] == '.fits'):
    print ("The file \"%s\" is not a FITS file!" % file_fits)
    print ("Check the file name.")
    sys.exit ()

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

# date/time
```

```
now = datetime.datetime.now ().isoformat ()

# opening FITS file
with astropy.io.fits.open (file_fits) as hdu_list:
    # reading header information
    header = hdu_list[0].header
    # WCS information
    wcs = astropy.wcs.WCS (header)
    # reading image data
    data   = hdu_list[0].data

# extraction of information from FITS header
exptime      = header[keyword_exptime]
filter_name  = header[keyword_filter]
airmass      = header[keyword_airmass]

# sky coordinate
coord_sky = astropy.coordinates.SkyCoord (target_ra_deg, target_dec_deg, \
                                          unit='deg')

# conversion from sky coordinate into pixel coordinate
coord_pix = wcs.world_to_pixel (coord_sky)

# initial guess of target position
init_x = coord_pix[0]
init_y = coord_pix[1]

# region of sub-frame for centroid measurement
subframe_xmin = int (init_x - halfwidth)
subframe_xmax = int (init_x + halfwidth + 1)
subframe_ymin = int (init_y - halfwidth)
subframe_ymax = int (init_y + halfwidth + 1)

# extracting sub-frame for centroid measurement
subframe = data[subframe_ymin:subframe_ymax, subframe_xmin:subframe_xmax]

# sky subtraction
subframe_skysub = subframe - numpy.median (subframe)

# centroid measurement
(com_x, com_y) = photutils.centroids.centroid_com (subframe_skysub)

# PSF fitting
subframe_y, subframe_x = numpy.indices (subframe_skysub.shape)
psf_init = astropy.modeling.models.Gaussian2D (x_mean=com_x, y_mean=com_y)
fit = astropy.modeling.fitting.LevMarLSQFitter ()
psf_fitted = fit (psf_init, subframe_x, subframe_y, subframe_skysub, \
                  maxiter=1000)

# fitted PSF parameters
centre_x = psf_fitted.x_mean.value + subframe_xmin
centre_y = psf_fitted.y_mean.value + subframe_ymin
theta    = psf_fitted.theta.value
fwhm_x   = psf_fitted.x_fwhm
fwhm_y   = psf_fitted.y_fwhm
fwhm     = (fwhm_x + fwhm_y) / 2.0

# position of centre of star in pixel coordinate
position_pix = (centre_x, centre_y)
```

```
# aperture radius in pixel
aperture_radius_pix  = fwhm * aperture_radius_fwhm
skyannulus_inner_pix = fwhm * skyannulus_inner_fwhm
skyannulus_outer_pix = fwhm * skyannulus_outer_fwhm

# making aperture
apphot_aperture \
    = photutils.aperture.CircularAperture (position_pix, r=aperture_radius_pix)
apphot_annulus \
    = photutils.aperture.CircularAnnulus (position_pix, \
                                          r_in=skyannulus_inner_pix, \
                                          r_out=skyannulus_outer_pix)

# making masked data for sky annulus
skyannulus_data       = apphot_annulus.to_mask (method='center').multiply (data)
skyannulus_mask       = skyannulus_data <= 0.0
skyannulus_maskeddata = 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_maskeddata, \
                                         sigma=threshold, maxiters=maxiters, \
                                         cenfunc='median')
skybg_per_pix = skybg_median

# aperture photometry
noise        = numpy.sqrt (data)
phot_star    = photutils.aperture.aperture_photometry (data, apphot_aperture, \
                                                       error=noise)
net_flux     = phot_star['aperture_sum'] - skybg_per_pix * apphot_aperture.area
net_flux_err = phot_star['aperture_sum_err']

# instrumental magnitude
instmag     = -2.5 * numpy.log10 (net_flux / exptime)
instmag_err = 2.5 / numpy.log (10) * net_flux_err / net_flux

# writing results into a file
with open (file_output, 'w') as fh:
    fh.write ("#\n")
    fh.write ("# Result of Aperture Photometry\n")
    fh.write ("#\n")
    fh.write ("#  Date/Time of Analysis\n")
    fh.write ("#   Date/Time = %s\n" % now)
    fh.write ("#\n")
    fh.write ("#  Input Parameters\n")
    fh.write ("#   FITS file                    = %s\n" % file_fits)
    fh.write ("#   RA                           = %f deg\n" % target_ra_deg)
    fh.write ("#   Dec                          = %f deg\n" % target_dec_deg)
    fh.write ("#   aperture radius              = %f in FWHM\n" \
              % aperture_radius_fwhm)
    fh.write ("#   inner sky annulus            = %f in FWHM\n" \
              % skyannulus_inner_fwhm)
    fh.write ("#   outer sky annulus            = %f in FWHM\n" \
              % skyannulus_outer_fwhm)
    fh.write ("#   half-width for centroid      = %f pixel\n" % halfwidth)
    fh.write ("#   threshold for sigma-clipping = %f in sigma\n" % threshold)
    fh.write ("#   number of max iterations     = %d\n" % maxiters)
    fh.write ("#   keyword for airmass          = %s\n" % keyword_airmass)
```

```
    fh.write ("#   keyword for exposure time    = %s\n" % keyword_exptime)
    fh.write ("#   keyword for filter name      = %s\n" % keyword_filter)
    fh.write ("#\n")
    fh.write ("#  Calculated and Measured Quantities\n")
    fh.write ("#   (init_x, init_y)     = (%f, %f)\n" % (init_x, init_y) )
    fh.write ("#   (com_x, com_y)       = (%f, %f)\n" \
              % (com_x + subframe_xmin, com_y + subframe_ymin) )
    fh.write ("#   (centre_x, centre_y) = (%f, %f)\n" % (centre_x, centre_y) )
    fh.write ("#   FWHM of stellar PSF  = %f pixel\n" % fwhm)
    fh.write ("#   sky background level = %f ADU per pixel\n" % skybg_per_pix)
    fh.write ("#   net flux             = %f ADU\n" % net_flux)
    fh.write ("#   net flux err         = %f ADU\n" % net_flux_err)
    fh.write ("#   instrumental mag     = %f\n" % instmag)
    fh.write ("#   instrumental mag err = %f\n" % instmag_err)
    fh.write ("#\n")
    fh.write ("#  Results\n")
    fh.write ("#   file, exptime, filter, centre_x, centre_y,\n"
              "#   net_flux, net_flux_err, instmag, instmag_err, airmass\n")
    fh.write ("%s %f %s %f %f %f %f %f %f %f\n" \
              % (file_fits, exptime, filter_name, centre_x, centre_y, \
                 net_flux, net_flux_err, instmag, instmag_err, airmass) )
```

Execute the script and check the results.

```
% chmod a+x ao2021_s15_06.py
% ./ao2021_s15_06.py -h
usage: ao2021_s15_06.py [-h] [-i INPUT] [-o OUTPUT] [-r RA] [-d DEC]
                        [-a APERTURE] [-w HALFWIDTH] [-s1 SKYANNULUS1]
                        [-s2 SKYANNULUS2] [-t THRESHOLD] [-n MAXITERS]
                        [-e KEYWORD_EXPTIME] [-f KEYWORD_FILTER]
                        [-m KEYWORD_AIRMASS]

aperture photometry of a star at given RA and Dec

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input FITS file name
  -o OUTPUT, --output OUTPUT
                        output file name
  -r RA, --ra RA        RA in degree
  -d DEC, --dec DEC     Dec in degree
  -a APERTURE, --aperture APERTURE
                        aperture radius in FWHM (default: 1.5)
  -w HALFWIDTH, --halfwidth HALFWIDTH
                        half-width for centroid measurement (default: 10)
  -s1 SKYANNULUS1, --skyannulus1 SKYANNULUS1
                        inner sky annulus radius in FWHM (default: 4)
  -s2 SKYANNULUS2, --skyannulus2 SKYANNULUS2
                        outer sky annulus radius in FWHM (default: 7)
  -t THRESHOLD, --threshold THRESHOLD
                        threshold for sigma-clipping in sigma (default: 3)
  -n MAXITERS, --maxiters MAXITERS
                        maximum number of iterations (default: 10)
  -e KEYWORD_EXPTIME, --keyword-exptime KEYWORD_EXPTIME
                        FITS keyword for exposure time (default: EXPTIME)
  -f KEYWORD_FILTER, --keyword-filter KEYWORD_FILTER
                        FITS keyword for filter name (default: FILTER)
```

```
  -m KEYWORD_AIRMASS, --keyword-airmass KEYWORD_AIRMASS
                        FITS keyword for airmass (default: AIRMASS)

% ./ao2021_s15_06.py -i pg1047/lot_20210214_0145_df.fits \
? -r 162.59599 -d -0.0818 -o test.phot
WARNING: FITSFixedWarning: RADECSYS= 'FK5 ' / Equatorial coordinate system
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
/home/daisuke/tex/astro/NCU/Lecture/AdvObs_2020b/15_extinction/script_15/./ao202
1_s15_06.py:179: RuntimeWarning: invalid value encountered in sqrt
  noise        = numpy.sqrt (data)
% cat test.phot
#
# Result of Aperture Photometry
#
#  Date/Time of Analysis
#   Date/Time = 2021-05-11T23:58:00.362743
#
#  Input Parameters
#   FITS file                    = pg1047/lot_20210214_0145_df.fits
#   RA                           = 162.595990 deg
#   Dec                          = -0.081800 deg
#   aperture radius              = 1.500000 in FWHM
#   inner sky annulus            = 4.000000 in FWHM
#   outer sky annulus            = 7.000000 in FWHM
#   half-width for centroid      = 10.000000 pixel
#   threshold for sigma-clipping = 3.000000 in sigma
#   number of max iterations     = 10
#   keyword for airmass          = AIRMASS
#   keyword for exposure time    = EXPTIME
#   keyword for filter name      = FILTER
#
#  Calculated and Measured Quantities
#   (init_x, init_y)     = (559.368692, 1553.972579)
#   (com_x, com_y)       = (560.096644, 1553.945340)
#   (centre_x, centre_y) = (559.972303, 1554.000732)
#   FWHM of stellar PSF  = 4.410534 pixel
#   sky background level = 11.089612 ADU per pixel
#   net flux             = 52663.054374 ADU
#   net flux err         = 232.782986 ADU
#   instrumental mag     = -9.303765
#   instrumental mag err = 0.004799
#
#  Results
#   file, exptime, filter, centre_x, centre_y,
#   net_flux, net_flux_err, instmag, instmag_err, airmass
pg1047/lot_20210214_0145_df.fits 10.000000 gp_Astrodon_2019 559.972303 1554.0007
32 52663.054374 232.782986 -9.303765 0.004799 1.094255
```

3.4 Doing photometry for all the g'-band data

Make a Python script to generate a shell script to carry out photometry for all the g'-band data except 180-sec data.

Python Code 7: ao2021_s15_07.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 = "checking FITS header"
parser = argparse.ArgumentParser (description=desc)

# adding argument
parser.add_argument ('-f', '--filter', default='', help='filter name')
parser.add_argument ('-n', '--name', default='', help='name of target object')
parser.add_argument ('-e1', '--exptime-min', type=float, default=5.0, \
                     help='minimum exposure time for use')
parser.add_argument ('-e2', '--exptime-max', type=float, default=90.0, \
                     help='maximum exposure time for use')
parser.add_argument ('-a', '--aperture', type=float, default=1.5, \
                     help='aperture radius in FWHM (default: 1.5)')
parser.add_argument ('-r', '--ra', type=float, default=-999.999, \
                     help='RA in degree')
parser.add_argument ('-d', '--dec', type=float, default=-999.999, \
                     help='Dec in degree')
parser.add_argument ('files', nargs='+', help='FITS files')

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

# input parameters
target_name          = args.name
filter_name          = args.filter
exptime_min          = args.exptime_min
exptime_max          = args.exptime_max
aperture_radius_fwhm = args.aperture
target_ra_deg        = args.ra
target_dec_deg       = args.dec
files_fits           = args.files

# checking "target_name", and "filter_name"
if (target_name == ''):
    print ("You have to specify target object name by using -n option!")
    sys.exit ()
if (filter_name == ''):
    print ("You have to specify filter name by using -f option!")
    sys.exit ()

# processing each FITS file
for file_fits in files_fits:
    # file name
    filename = file_fits.split ('/') [-1]
    name_list = file_fits.split ('_')
    frame_id = int (name_list[-2])
    file_output = "phot_%s_%04d.data" % (filter_name, frame_id)
    # if the file is not a FITS file, then skip
    if not (file_fits[-5:] == '.fits'):
        continue
```

```
    # if the file is not a reduced data, then skip
    if not (file_fits[-8:] == '_df.fits'):
        continue
    # opening a FITS file
    with astropy.io.fits.open (file_fits) as hdu_list:
        # reading header information
        header = hdu_list[0].header
    # checking header information
    if not ( (header['IMAGETYP'] == 'LIGHT') \
             and (header['OBJECT'] == target_name) \
             and (header['FILTER'] == filter_name) \
             and (header['EXPTIME'] > exptime_min) \
             and (header['EXPTIME'] < exptime_max) ):
        continue
    # printing command
    print ("./ao2021_s15_06.py -i %s -r %f -d %f -a %f -o %s" \
           % (file_fits, target_ra_deg, target_dec_deg, \
              aperture_radius_fwhm, file_output) )
```

Run the script and generate a shell script for photometry.

```
% chmod a+x ao2021_s15_07.py
% ./ao2021_s15_07.py -h
usage: ao2021_s15_07.py [-h] [-f FILTER] [-n NAME] [-e1 EXPTIME_MIN]
                        [-e2 EXPTIME_MAX] [-a APERTURE] [-r RA] [-d DEC]
                        files [files ...]

checking FITS header

positional arguments:
  files                 FITS files

optional arguments:
  -h, --help            show this help message and exit
  -f FILTER, --filter FILTER
                        filter name
  -n NAME, --name NAME  name of target object
  -e1 EXPTIME_MIN, --exptime-min EXPTIME_MIN
                        minimum exposure time for use
  -e2 EXPTIME_MAX, --exptime-max EXPTIME_MAX
                        maximum exposure time for use
  -a APERTURE, --aperture APERTURE
                        aperture radius in FWHM (default: 1.5)
  -r RA, --ra RA        RA in degree
  -d DEC, --dec DEC     Dec in degree

% ./ao2021_s15_07.py -r 162.59599 -d -0.0818 -f gp_Astrodon_2019 \
? -n PG1047 pg1047/*.fits > phot_g.sh
% head -5 phot_g.sh
./ao2021_s15_06.py -i pg1047/lot_20210214_0145_df.fits -r 162.595990 -d -0.08180
0 -a 1.500000 -o phot_gp_Astrodon_2019_0145.data
./ao2021_s15_06.py -i pg1047/lot_20210214_0146_df.fits -r 162.595990 -d -0.08180
0 -a 1.500000 -o phot_gp_Astrodon_2019_0146.data
./ao2021_s15_06.py -i pg1047/lot_20210214_0147_df.fits -r 162.595990 -d -0.08180
0 -a 1.500000 -o phot_gp_Astrodon_2019_0147.data
./ao2021_s15_06.py -i pg1047/lot_20210214_0245_df.fits -r 162.595990 -d -0.08180
0 -a 1.500000 -o phot_gp_Astrodon_2019_0245.data
./ao2021_s15_06.py -i pg1047/lot_20210214_0246_df.fits -r 162.595990 -d -0.08180
```

```
0 -a 1.500000 -o phot_gp_Astrodon_2019_0246.data
```

Execute the shell script.

```
% sh phot_g.sh
% ls phot_gp_Astrodon_2019_*.data
phot_gp_Astrodon_2019_0145.data         phot_gp_Astrodon_2019_0286.data
phot_gp_Astrodon_2019_0146.data         phot_gp_Astrodon_2019_0325.data
phot_gp_Astrodon_2019_0147.data         phot_gp_Astrodon_2019_0326.data
phot_gp_Astrodon_2019_0245.data         phot_gp_Astrodon_2019_0327.data
phot_gp_Astrodon_2019_0246.data         phot_gp_Astrodon_2019_0366.data
phot_gp_Astrodon_2019_0247.data         phot_gp_Astrodon_2019_0367.data
phot_gp_Astrodon_2019_0285.data         phot_gp_Astrodon_2019_0368.data
% cat phot_gp_Astrodon_2019_* | grep -v # > phot_g.data
% head -5 phot_g.data
pg1047/lot_20210214_0145_df.fits 10.000000 gp_Astrodon_2019 559.972303 1554.0007
32 52663.054374 232.782986 -9.303765 0.004799 1.094255
pg1047/lot_20210214_0146_df.fits 10.000000 gp_Astrodon_2019 560.321805 1554.7207
11 52031.703515 231.144157 -9.290670 0.004823 1.094421
pg1047/lot_20210214_0147_df.fits 10.000000 gp_Astrodon_2019 559.708725 1554.6743
18 52195.081419 231.123438 -9.294074 0.004808 1.094585
pg1047/lot_20210214_0245_df.fits 10.000000 gp_Astrodon_2019 550.916839 1552.8455
02 50695.388761 229.087372 -9.262421 0.004906 1.277339
pg1047/lot_20210214_0246_df.fits 10.000000 gp_Astrodon_2019 550.865047 1552.8113
57 50858.344863 229.802222 -9.265906 0.004906 1.278867
```

3.5 Determining g'-band atmospheric extinction coefficient

Make a Python script to calculate g'-band atmospheric extinction coefficient using least-squares fitting.

Python Code 8: ao2021_s15_08.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# import numpy module
import numpy

# importing scipy module
import scipy.optimize

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

# constructing parser object
desc = "determining atmospheric extinction coefficient"
parser = argparse.ArgumentParser (description=desc)

# adding argument
parser.add_argument ('-i', '--input', default='', help='input data file')
parser.add_argument ('-o', '--output', default='', help='output image file')
```

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

# input parameters
file_input  = args.input
file_output = args.output

# checking file_input and file_output
if ( (file_input == '') or (file_output == '') ):
    print ("Input and output file names must be given.")
    sys.exit ()
if not ( (file_output[-4:] == '.eps') or (file_output[-4:] == '.pdf') \
         or (file_output[-4:] == '.png') or (file_output[-3:] == '.ps') ):
    print ("Output image file must be either EPS, PDF, PNG, or PS.")
    print ("Given output image file name = %s" % file_output)
    sys.exit ()

# making empty numpy arrays for data
array_instmag     = numpy.array ([])
array_instmag_err = numpy.array ([])
array_airmass     = numpy.array ([])

# opening input data file
with open (file_input, 'r') as fh:
    # reading file line-by-line
    for line in fh:
        # splitting line
        data = line.split ()
        # instmag, instmag_err, airmass
        instmag     = float (data[7])
        instmag_err = float (data[8])
        airmass     = float (data[9])
        # adding data to numpy arrays
        array_instmag     = numpy.append (array_instmag, instmag)
        array_instmag_err = numpy.append (array_instmag_err, instmag_err)
        array_airmass     = numpy.append (array_airmass, airmass)

# least-squares method

# initial values of coefficients of fitted function
a = 1.0
b = 1.0

# function for least-squares fitting
def func (x, a, b):
    y = a * x + b
    return y

# least-squares fitting
popt, pcov = scipy.optimize.curve_fit (func, array_airmass, array_instmag, \
                                       p0=(a,b), sigma=array_instmag_err)

# fitted coefficients
print ("popt:")
print (popt)

# covariance matrix
print ("pcov:")
```

```
print (pcov)

# fitted a and b
a_fitted = popt[0]
b_fitted = popt[1]

# degree of freedom
dof = len (array_airmass) - 2
print ("dof =", dof)

# residual
residual = array_instmag - func (array_airmass, a_fitted, b_fitted)
reduced_chi2 = (residual**2).sum () / dof
print ("reduced chi^2 =", reduced_chi2)

# errors of a and b
a_err = numpy.sqrt (pcov[0][0])
b_err = numpy.sqrt (pcov[1][1])
print ("a = %f +/- %f (%f %%)" % (a_fitted, a_err, a_err / a_fitted * 100.0) )
print ("b = %f +/- %f (%f %%)" % (b_fitted, b_err, b_err / b_fitted * 100.0) )

# fitted line
fitted_x = numpy.linspace (1.0, 2.5, 10**6)
fitted_y = a_fitted * fitted_x + b_fitted

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

# axes
ax.set_xlabel ("Airmass")
ax.set_ylabel ("Instrumental Magnitude [mag]")
ax.invert_yaxis ()

# plotting image
ax.plot (fitted_x, fitted_y, 'c--', label='least-squares fitting')
ax.errorbar (array_airmass, array_instmag, yerr=array_instmag_err, \
             fmt='ro', ecolor='black', capsize=5, \
             label='photometric standard star')
ax.legend ()

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

Execute the script, and determine g'-band atmospheric extinction coefficient. (Fig. 2)

```
% chmod a+x ao2021_s15_08.py
% ./ao2021_s15_08.py -h
usage: ao2021_s15_08.py [-h] [-i INPUT] [-o OUTPUT]

determining atmospheric extinction coefficient

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input data file
  -o OUTPUT, --output OUTPUT
```

```
                        output image file

% ./ao2021_s15_08.py -i phot_g.data -o extinction_g.pdf
popt:
[ 0.10776497 -9.40715029]
pcov:
[[ 5.15728100e-05 -7.49613454e-05]
 [-7.49613454e-05  1.12578965e-04]]
dof = 12
reduced chi^2 = 8.069726288518303e-05
a = 0.107765 +/- 0.007181 (6.663966 %)
b = -9.407150 +/- 0.010610 (-0.112790 %)
% ls -l extinction_g.pdf
-rw-r--r--  1 daisuke  taiwan  16264 May 12 01:40 extinction_g.pdf
% xpdf extinction_g.pdf
```

The g'-band atmospheric extinction coefficient is estimated to be 0.108 ± 0.007 mag/airmass.

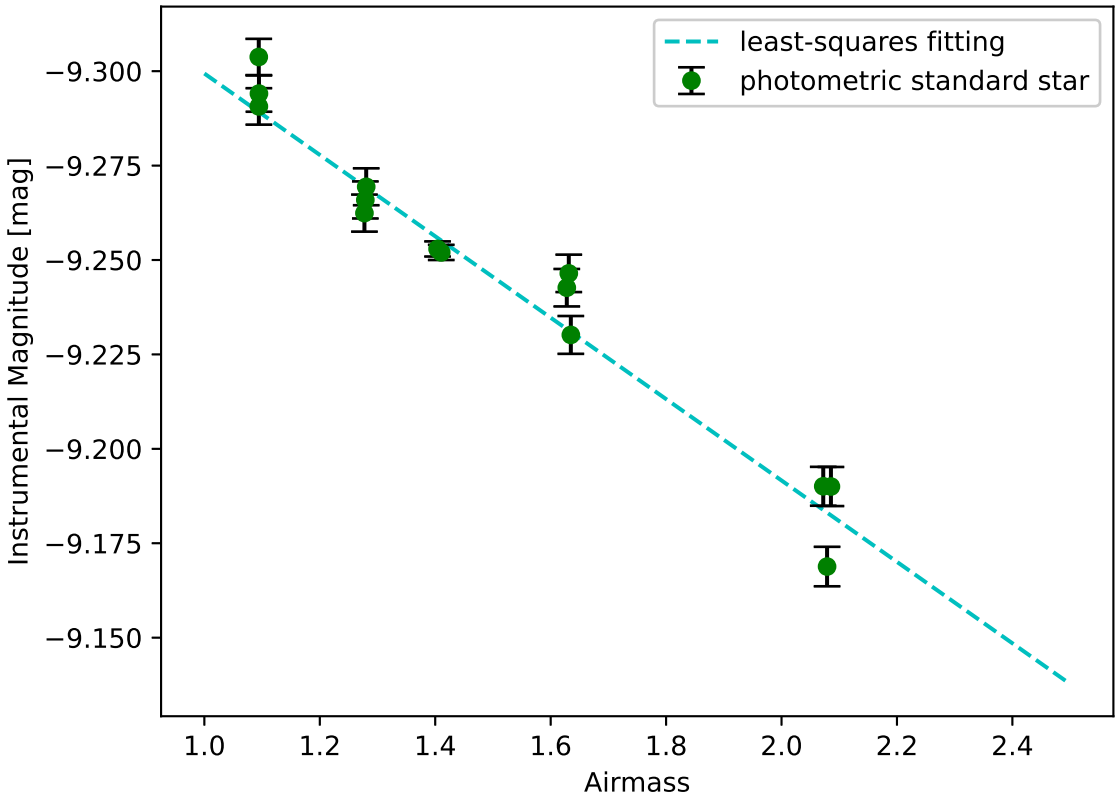

Figure 2: Atmospheric extinction at SDSS g'-band. The atmospheric extinction coefficient at g'-band is 0.108 ± 0.007 mag/airmass.

4 For your training

1. Read the textbook "Astronomical Photometry" to learn about atmospheric extinction.
   - "Astronomical Photometry"

    - Arne A. Henden, Ronald H. Kaitchuck
    - `https://ui.adsabs.harvard.edu/abs/1982asph.book.....H/abstract`

2. Read section 9 and 19 of the document "An Introduction to Astronomical Photometry using CCDs" and learn about atmospheric extinction.

    - An Introduction to Astronomical Photometry using CCDs
        - William Romanishin
        - `http://hildaandtrojanasteroids.net/wrccd22oct06.pdf`

3. Read section 8 of "The CCD Photometric Calibration Cookbook" to learn about atmospheric extinction.

    - `https://starlink.rl.ac.uk/star/docs/sc6.pdf`

5 Assignment

1. What is airmass? Describe about it. How can we calculate airmass?
2. What is atmospheric extinction? Describe about it. How to determine atmospheric extinction coefficients?
3. Atmospheric extinction coefficients
    (a) What is typical value of U-band atmospheric extinction coefficient? Show the references you have gathered.
    (b) What is typical value of B-band atmospheric extinction coefficient? Show the references you have gathered.
    (c) What is typical value of V-band atmospheric extinction coefficient? Show the references you have gathered.
    (d) What is typical value of R-band atmospheric extinction coefficient? Show the references you have gathered.
    (e) What is typical value of I-band atmospheric extinction coefficient? Show the references you have gathered.
    (f) What is typical value of u'-band atmospheric extinction coefficient? Show the references you have gathered.
    (g) What is typical value of g'-band atmospheric extinction coefficient? Show the references you have gathered.
    (h) What is typical value of r'-band atmospheric extinction coefficient? Show the references you have gathered.
    (i) What is typical value of i'-band atmospheric extinction coefficient? Show the references you have gathered.
    (j) What is typical value of z'-band atmospheric extinction coefficient? Show the references you have gathered.
4. Estimate the value r'-band atmospheric extinction coefficient from PG 1047+003 field data taken on 14 February 2021. Use the star ID 8 of PG 1047+003 field. Describe what you have done. Show the source code of all the Python script you have written. Show the value of atmospheric extinction coefficient.
5. Estimate the value i'-band atmospheric extinction coefficient from PG 1047+003 field data taken on 14 February 2021. Use the star ID 8 of PG 1047+003 field. Describe what you have done. Show the source code of all the Python script you have written. Show the value of atmospheric extinction coefficient.
6. Estimate the value g'-band atmospheric extinction coefficient from PG 1047+003 field data taken on 14 February 2021. Use the star other than star ID 8 of PG 1047+003 field. Describe what you have done. Show the source code of all the Python script you have written. Show the value of atmospheric extinction coefficient.
7. Estimate the value r'-band atmospheric extinction coefficient from PG 1047+003 field data taken on 14 February 2021. Use the star other than star ID 8 of PG 1047+003 field. Describe what you have done. Show the source code of all the Python script you have written. Show the value of atmospheric extinction coefficient.
8. Estimate the value i'-band atmospheric extinction coefficient from PG 1047+003 field data taken on 14 February 2021. Use the star other than star ID 8 of PG 1047+003 field. Describe what you have done. Show the source code of all the Python script you have written. Show the value of atmospheric extinction coefficient.