Advanced Astronomical Observations 2022
Session 13: Atmospheric Extinction

Kinoshita Daisuke

29 April 2022
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 2022 to June 2022.
  - 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 Data for this session

We process the data of the standard star field PG1047+003. If you have finished data reduction for the data taken on 14/Feb/2021, then copy reduced data of PG1047+003 field.

If you have not yet finished data reduction for the data taken on 14/Feb/2021, then download reduced data.

About the data reduction for the data taken on 14/Feb/2021, check the course material for the session 09 "Basic CCD Data reduction 2".

1.1 Copying the data

Check the data taken on 14 February 2021, and search for reduced FITS files of the standard star field PG1047+003.

Python Code 1: advobs202202_s13_01.py

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

#
# Time-stamp: <2022/04/27 22:11:46 (CST) daisuke>
#

# 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='reduced 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
list_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 list_fits:
    # file name
    file_fits = str (path_fits)

    # if the file is not a FITS file, then skip
    if not (path_fits.suffix == '.fits'):
        # printing message
        print ("# file '%s' is not a FITS file, skipping..." % file_fits)
        # skipping to next
        continue
    # if the file is not a reduced data, then skip
    if not (path_fits.stem[-3:] == '_df'):
        # printing message
        print ("# file '%s' is not a reduced data, skipping..." % file_fits)
```

```
        # skipping to next
        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
    # data type should be 'LIGHT' (= object frame)
    # target object must be matched with the specified name
    if not ( (header['IMAGETYP'] == 'LIGHT') \
             and (header['OBJECT'] == target_name) ):
        # if not matched, skip
        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 PG1047+003. Replace the directory name "`../../s09/script_09/red`" with the directory where you have reduced data.

```
% chmod a+x advobs202202_s13_01.py
% ./advobs202202_s13_01.py -h
usage: advobs202202_s13_01.py [-h] [-d DIR] [-n NAME]

checking FITS header

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

% ./advobs202202_s13_01.py -d ../../s09/script_09/red -n PG1047
# file name, time-obs, data type, exptime, filter, object, airmass
# file '../../s09/script_09/red/dark_00005000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/dark_00010000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/dark_00015000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/dark_00045000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/dark_00060000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/dark_00180000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/flat_gp_Astrodon_2019.fits' is not a reduced dat
a, skipping...
# file '../../s09/script_09/red/flat_ip_Astrodon_2019.fits' is not a reduced dat
a, skipping...
# file '../../s09/script_09/red/flat_rp_Astrodon_2019.fits' is not a reduced dat
a, skipping...
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
% ./advobs202202_s13_01.py -d ../../s09/script_09/red -n PG1047 | grep -v # | wc
      51     357    3978
```

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

Python Code 2: advobs202202_s13_02.py

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

#
```

```
# Time-stamp: <2022/04/27 22:36:14 (CST) daisuke>
#

# 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='reduced data directory')
parser.add_argument ('-n', '--name', default='', help='name of target object')
parser.add_argument ('-t', '--to', default='', help='target directory name')

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

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

# checking "dir_data" and "target_name"
if (dir_data == ''):
    print ("You have to specify data directory name by using -d option!")
    sys.exit ()
if (dir_destination == ''):
    print ("You have to specify destination directory name by using -t 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 ()

# making a pathlib object for "dir_destination"
path_destination = pathlib.Path (dir_destination)

# if directory does not exist, then make a directory
if not (path_destination.exists ()):
    # making a directory
```

```
    path_destination.mkdir (parents=True, exist_ok=True)

# reading directory
list_fits = path_datadir.iterdir ()

# processing each FITS file
for path_fits in list_fits:
    # file name
    file_fits = str (path_fits)

    # if the file is not a FITS file, then skip
    if not (path_fits.suffix == '.fits'):
        # printing message
        print ("# file '%s' is not a FITS file, skipping..." % file_fits)
        # skipping to next
        continue
    # if the file is not a reduced data, then skip
    if not (path_fits.stem[-3:] == '_df'):
        # printing message
        print ("# file '%s' is not a reduced data, skipping..." % file_fits)
        # skipping to next
        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
    # data type should be 'LIGHT' (= object frame)
    # target object must be matched with the specified name
    if not ( (header['IMAGETYP'] == 'LIGHT') \
             and (header['OBJECT'] == target_name) ):
        # if not matched, skip
        continue

    # printing status
    print ("Now copying the file '%s' to '%s'..." \
           % (path_fits.name, dir_destination) )

    # copying the file to current directory
    shutil.copy2 (file_fits, path_destination)

    # printing status
    print ("Finished copying the file '%s' to '%s'!" \
           % (path_fits.name, dir_destination) )
```

Run the script, and copy files.

```
% chmod a+x advobs202202_s13_02.py
% ./advobs202202_s13_02.py -h
usage: advobs202202_s13_02.py [-h] [-d DIR] [-n NAME] [-t TO]

copying FITS files

optional arguments:
  -h, --help            show this help message and exit
  -d DIR, --dir DIR     reduced data directory
```

```
  -n NAME, --name NAME  name of target object
  -t TO, --to TO        target directory name

% ./advobs202202_s13_02.py -d ../../s09/script_09/red -n PG1047 -t pg1047
# file '../../s09/script_09/red/dark_00005000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/dark_00010000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/dark_00015000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/dark_00045000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/dark_00060000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/dark_00180000.fits' is not a reduced data, skipp
ing...
# file '../../s09/script_09/red/flat_gp_Astrodon_2019.fits' is not a reduced dat
a, skipping...
# file '../../s09/script_09/red/flat_ip_Astrodon_2019.fits' is not a reduced dat
a, skipping...
# file '../../s09/script_09/red/flat_rp_Astrodon_2019.fits' is not a reduced dat
a, skipping...
Now copying the file 'lot_20210214_0145_df.fits' to 'pg1047'...
Finished copying the file 'lot_20210214_0145_df.fits' to 'pg1047'!
Now copying the file 'lot_20210214_0146_df.fits' to 'pg1047'...
Finished copying the file 'lot_20210214_0146_df.fits' to 'pg1047'!
Now copying the file 'lot_20210214_0147_df.fits' to 'pg1047'...
Finished copying the file 'lot_20210214_0147_df.fits' to 'pg1047'!
Now copying the file 'lot_20210214_0148_df.fits' to 'pg1047'...
Finished copying the file 'lot_20210214_0148_df.fits' to 'pg1047'!
Now copying the file 'lot_20210214_0149_df.fits' to 'pg1047'...
Finished copying the file 'lot_20210214_0149_df.fits' to 'pg1047'!

.....

Now copying the file 'lot_20210214_0370_df.fits' to 'pg1047'...
Finished copying the file 'lot_20210214_0370_df.fits' to 'pg1047'!
Now copying the file 'lot_20210214_0371_df.fits' to 'pg1047'...
Finished copying the file 'lot_20210214_0371_df.fits' to 'pg1047'!
Now copying the file 'lot_20210214_0372_df.fits' to 'pg1047'...
Finished copying the file 'lot_20210214_0372_df.fits' to 'pg1047'!
Now copying the file 'lot_20210214_0373_df.fits' to 'pg1047'...
Finished copying the file 'lot_20210214_0373_df.fits' to 'pg1047'!
Now copying the file 'lot_20210214_0374_df.fits' to 'pg1047'...
Finished copying the file 'lot_20210214_0374_df.fits' to 'pg1047'!
% ls pg1047/
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_0150_df.fits               lot_20210214_0292_df.fits
lot_20210214_0151_df.fits               lot_20210214_0293_df.fits
lot_20210214_0152_df.fits               lot_20210214_0325_df.fits
lot_20210214_0153_df.fits               lot_20210214_0326_df.fits
lot_20210214_0185_df.fits               lot_20210214_0327_df.fits
lot_20210214_0186_df.fits               lot_20210214_0328_df.fits
lot_20210214_0187_df.fits               lot_20210214_0329_df.fits
lot_20210214_0188_df.fits               lot_20210214_0330_df.fits
```

```
lot_20210214_0189_df.fits                lot_20210214_0331_df.fits
lot_20210214_0190_df.fits                lot_20210214_0332_df.fits
lot_20210214_0245_df.fits                lot_20210214_0333_df.fits
lot_20210214_0246_df.fits                lot_20210214_0366_df.fits
lot_20210214_0247_df.fits                lot_20210214_0367_df.fits
lot_20210214_0248_df.fits                lot_20210214_0368_df.fits
lot_20210214_0249_df.fits                lot_20210214_0369_df.fits
lot_20210214_0250_df.fits                lot_20210214_0370_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
% ls pg1047/*.fits | wc
      51      51    1683
```

1.2 Downloading and extracting the data

If you have not yet finished data reduction of the data taken on 14/Feb/2021, then download reduced data.

```
% curl -k -o pg1047.tar.xz \
? https://s3b.astro.ncu.edu.tw/advobs_202202/data/pg1047.tar.xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1433M  100 1433M    0     0  2054k       0  0:11:54  0:11:54 --:--:-- 4209k
```

Then, extract the data.

```
% tar xJvf pg1047.tar.xz
x pg1047/
x pg1047/lot_20210214_0145_df.fits
x pg1047/lot_20210214_0146_df.fits
x pg1047/lot_20210214_0147_df.fits
x pg1047/lot_20210214_0148_df.fits
x pg1047/lot_20210214_0149_df.fits
x pg1047/lot_20210214_0150_df.fits
x pg1047/lot_20210214_0151_df.fits
x pg1047/lot_20210214_0152_df.fits
x pg1047/lot_20210214_0153_df.fits
x pg1047/lot_20210214_0185_df.fits

.....

x pg1047/lot_20210214_0333_df.fits
x pg1047/lot_20210214_0366_df.fits
x pg1047/lot_20210214_0367_df.fits
x pg1047/lot_20210214_0368_df.fits
x pg1047/lot_20210214_0369_df.fits
x pg1047/lot_20210214_0370_df.fits
x pg1047/lot_20210214_0371_df.fits
x pg1047/lot_20210214_0372_df.fits
x pg1047/lot_20210214_0373_df.fits
x pg1047/lot_20210214_0374_df.fits
% ls pg1047/*.fits | wc
      51      51    1683
% ls pg1047
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
```

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  28481      0  0:00:02  0:00:02 --:--:-- 28485
% ls -lF stds.tar.gz
-rw-r--r--  1 daisuke  taiwan  65774 Apr 27 22:43 stds.tar.gz
```

2.2 Extracting standard star catalogue

```
% 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 105_a.txt.clean.v3
x 107_a.txt.clean.v3
x 107_b.txt.clean.v3
x 108_a.txt.clean.v3
x 108_b.txt.clean.v3

.....

x Ross374.txt.clean.v3
x Ross49.txt.clean.v3
x Ross530.txt.clean.v3
x Ross683.txt.clean.v3
x Ross711.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 ..
```

2.3 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: advobs202202_s13_03.py

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

#
# Time-stamp: <2022/04/27 22:51:47 (CST) daisuke>
#

# importing argparse module
import argparse

# importing astropy module
import astropy.coordinates

# constructing parser object
desc   = "reading standard star information from file"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('files', nargs='+', help='files to read')

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

# list of data files
list_files = args.files

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

# processing files
for file_data in list_files:
    # 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 advobs202202_s13_03.py
% ./advobs202202_s13_03.py -h
usage: advobs202202_s13_03.py [-h] files [files ...]

reading standard star information from file

positional arguments:
  files       files to read

optional arguments:
  -h, --help  show this help message and exit

% ./advobs202202_s13_03.py stds/PG1047+003A.txt.clean.v3
# 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
```

Write the output of the script into a file.

```
% ./advobs202202_s13_03.py stds/PG1047+003A.txt.clean.v3 > pg1047.txt
```

```
% cat pg1047.txt
# 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
% ls -lF pg1047.txt
-rw-r--r--  1 daisuke  taiwan  1085 Apr 27 22:54 pg1047.txt
% grep -v # pg1047.txt | wc
      13     104    1014
```

We have found 13 standard stars in the field of PG1047+003.

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: advobs202202_s13_04.py

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

#
# Time-stamp: <2022/04/27 23:28:50 (CST) daisuke>
#

# 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
list_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 list_fits:
    # file name
    file_fits = str (path_fits)

    # if the file is not a FITS file, then skip
    if not (path_fits.suffix == '.fits'):
        # printing message
        print ("# file '%s' is not a FITS file, skipping..." % file_fits)
        # skipping to next
        continue
    # if the file is not a reduced data, then skip
    if not (path_fits.stem[-3:] == '_df'):
        # printing message
        print ("# file '%s' is not a reduced data, skipping..." % file_fits)
        # skipping to next
        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 advobs202202_s13_04.py
% ./advobs202202_s13_04.py -h
usage: advobs202202_s13_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

% ./advobs202202_s13_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
% ./advobs202202_s13_04.py -n PG1047 -d pg1047 -f gp_Astrodon_2019 | grep -v # | wc
      17     119    1326
```

17 files are found.

3.2 Identifying photometric standard star

Check the list of standard stars in PG1047+003 field again.

```
% cat pg1047.txt
# 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
```

We carry out aperture photometry for the star ID 8 in the field of PG1047+003. Make a Python script to mark the location of the star ID 8.

Python Code 5: advobs202202_s13_05.py

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

#
# Time-stamp: <2022/04/28 12:02:45 (CST) daisuke>
#

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# 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.figure
import matplotlib.backends.backend_agg
import matplotlib.patches

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

# colour maps
choices_cmap = ['viridis', 'plasma', 'inferno', 'magma', 'cividis', \
                'binary', 'gray', 'bone', 'pink', \
                'spring', 'summer', 'autumn', 'winter', \
                'cool', 'hot', 'copper', 'ocean', 'terrain', \
                'gnuplot', 'cubehelix', 'jet', 'turbo']

# 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 (default: 10)')
parser.add_argument ('-w', '--width', type=float, default=2.0, \
                      help='width of circle (default: 2)')
parser.add_argument ('-l', '--resolution', type=int, default=450, \
                      help='resolution of output image in DPI (default: 450)')
parser.add_argument ('-c', '--cmap', default='bone', choices=choices_cmap, \
                      help='choice of colour map (default: bone)')

# 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
width          = args.width
resolution     = args.resolution
cmap           = args.cmap

# 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) ):
    # printing message
    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)
    # exit
    sys.exit ()

# making pathlib objects
path_fits   = pathlib.Path (file_fits)
path_output = pathlib.Path (file_output)

# existence checks
if not (path_fits.exists ()):
    # printing message
    print ("The file \"%s\" does not exist!")
    # exit
    sys.exit ()
if (path_output.exists ()):
    # printing message
    print ("The file \"%s\" exists!")
    # exit
    sys.exit ()

# check of FITS file name
if not (path_fits.suffix == '.fits'):
    # printing message
```

```
    print ("The file \"%s\" is not a FITS file!" % file_fits)
    print ("Check the file name.")
    # exit
    sys.exit ()

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

# 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

# coordinate of standard star in RA and Dec
coord = astropy.coordinates.SkyCoord (target_ra_deg, target_dec_deg, unit='deg')

# conversion into hmsdms format
radec_str = coord.to_string ('hmsdms')
(ra_str, dec_str) = radec_str.split ()

# (x, y) coordinate of standard star on image
(x, y) = wcs.world_to_pixel (coord)

# printing (RA, Dec) and (x, y) of standard star
print ("(RA, Dec) = (%s, %s)" % (ra_str, dec_str) )
print ("(x, y)    = (%f, %f)" % (x, y) )

# radius of circle in pixel
pix_scale    = abs (header['CDELT1']) * 3600.0
radius_pixel = radius_arcsec / pix_scale

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

# axes
ax.set_xlabel ('RA')
ax.set_ylabel ('Dec')

# plotting image
norm \
    = astropy.visualization.mpl_normalize.ImageNormalize \
    ( stretch=astropy.visualization.HistEqStretch (data) )
im = ax.imshow (data, origin='lower', cmap=cmap, norm=norm)
fig.colorbar (im)

# making a circle to indicate the location of standard star
stdstars = matplotlib.patches.Circle (xy=(x, y), radius=radius_pixel, \
```

```
                                        fill=False, color="red", linewidth=width)
# plotting location of standard star
ax.add_patch (stdstars)

# invert Y-axis
ax.invert_yaxis ()

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

Run the script, and make an image indicating the location of the star ID 8. The coordinate of the star ID 8 is $(RA, Dec) = (162.59599, -0.0818)$

```
% chmod a+x advobs202202_s13_05.py
% ./advobs202202_s13_05.py -h
usage: advobs202202_s13_05.py [-h] [-i INPUT] [-o OUTPUT] [-r RA] [-d DEC]
                              [-s SIZE] [-w WIDTH] [-l RESOLUTION]
                              [-c {viridis,plasma,inferno,magma,cividis,binary,g
ray,bone,pink,spring,summer,autumn,winter,cool,hot,copper,ocean,terrain,gnuplot,
cubehelix,jet,turbo}]

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 (default: 10)
  -w WIDTH, --width WIDTH
                        width of circle (default: 2)
  -l RESOLUTION, --resolution RESOLUTION
                        resolution of output image in DPI (default: 450)
  -c {viridis,plasma,inferno,magma,cividis,binary,gray,bone,pink,spring,summer,a
utumn,winter,cool,hot,copper,ocean,terrain,gnuplot,cubehelix,jet,turbo}, --cmap
{viridis,plasma,inferno,magma,cividis,binary,gray,bone,pink,spring,summer,autumn
,winter,cool,hot,copper,ocean,terrain,gnuplot,cubehelix,jet,turbo}
                        choice of colour map (default: bone)

% ./advobs202202_s13_05.py -r 162.59599 -d -0.0818 -c viridis -s 20 \
? -i pg1047/lot_20210214_0145_df.fits -o pg1047_0145_008.pdf
WARNING: FITSFixedWarning: RADECSYS= 'FK5 ' / Equatorial coordinate system
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59259.725532
 from DATE-OBS'. [astropy.wcs.wcs]
(RA, Dec) = (10h50m23.0376s, -00d04m54.48s)
(x, y)    = (559.368692, 1553.972579)
% ls -lF *.pdf
-rw-r--r--  1 daisuke  taiwan  2706947 Apr 28 11:59 pg1047_0145_008.pdf
```

The star ID 8 in the field of PG 1047+003 is located at (x, y) = ($\sim$ 559, $\sim$ 1554). (Fig. 1)
Show the image. Now, we know which star is the star ID 8 in PG1047+003 field.

```
% okular pg1047_0145_008.pdf
```

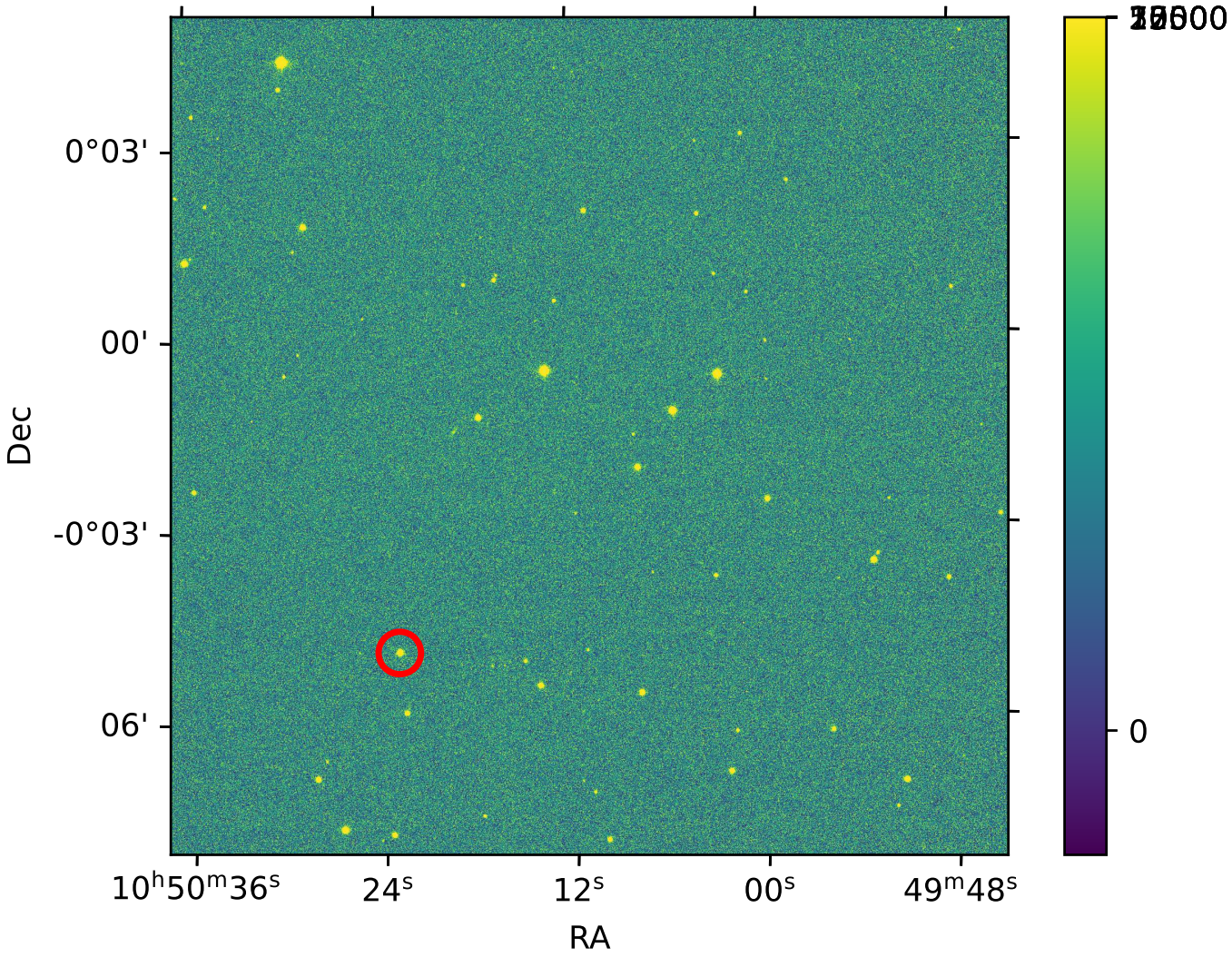

Figure 1: The location of the star ID 8 in the field of PG1047+003.

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: advobs202202_s13_06.py

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

#
# Time-stamp: <2022/04/28 14:00:55 (CST) daisuke>
#

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing datetime module
import datetime

# importing numpy module
import numpy

# importing astropy module
import astropy.coordinates
import astropy.stats

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

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

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

# centroid measurement technique
choices_centroid = ['com', '1dg', '2dg']

# PSF models (Gaussian and Moffat)
choices_psf = ['2dg', '2dm']

# colour maps
choices_cmap = ['viridis', 'plasma', 'inferno', 'magma', 'cividis', \
                'binary', 'gray', 'bone', 'pink', \
                'spring', 'summer', 'autumn', 'winter', \
                'cool', 'hot', 'copper', 'ocean', 'terrain', \
                'gnuplot', 'cubehelix', 'jet', 'turbo']

# adding argument
parser.add_argument ('-i', '--input', default='', \
                     help='input FITS file name')
```

```
parser.add_argument ('-o', '--output', default='', \
                     help='output data file name')
parser.add_argument ('-g', '--graphic', default='', \
                     help='output graphic file name')
parser.add_argument ('-c', '--centroid', choices=choices_centroid, \
                     default='2dg', \
                     help='centroid measurement algorithm (default: 2dg)')
parser.add_argument ('-p', '--psf', choices=choices_psf, default='2dg', \
                     help='PSF model [2dg=Gaussian, 2dm=Moffat] (default: 2dg)')
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 ('-a', '--aperture', type=float, default=2.0, \
                     help='aperture radius in FWHM (default: 2.0)')
parser.add_argument ('-w', '--halfwidth', type=int, default=20, \
                     help='half-width for centroid measurement (default: 20)')
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=4.0, \
                     help='threshold for sigma-clipping in sigma (default: 4)')
parser.add_argument ('-n', '--maxiters', type=int, default=100, \
                     help='maximum number of iterations (default: 100)')
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)')
parser.add_argument ('-z', '--cmap', default='bone', choices=choices_cmap, \
                     help='choice of colour map (default: bone)')
parser.add_argument ('-l', '--resolution', type=int, default=450, \
                     help='resolution in DPI (default: 450)')

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

# input parameters
file_fits             = args.input
file_output           = args.output
file_graphic          = args.graphic
centroid              = args.centroid
psf_model             = args.psf
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
cmap                  = args.cmap
resolution            = args.resolution
```

```
# 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) ):
    # printing message
    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)
    # exit
    sys.exit ()

# making pathlib objects
path_fits    = pathlib.Path (file_fits)
path_output  = pathlib.Path (file_output)
path_graphic = pathlib.Path (file_graphic)

# existence checks
if not (path_fits.exists ()):
    # printing message
    print ("The file \"%s\" does not exist!")
    # exit
    sys.exit ()
if (path_output.exists ()):
    # printing message
    print ("The file \"%s\" exists!" % file_output)
    # exit
    sys.exit ()
if (path_graphic.exists ()):
    # printing message
    print ("The file \"%s\" exists!" % file_graphic)
    # exit
    sys.exit ()

# check of FITS file name
if not (path_fits.suffix == '.fits'):
    # printing message
    print ("The file \"%s\" is not a FITS file!" % file_fits)
    print ("Check the file name.")
    # exit
    sys.exit ()

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

# check of output graphic file
if not ( (path_graphic.suffix == '.eps') or (path_graphic.suffix == '.pdf') \
         or (path_graphic.suffix == '.png') or (path_graphic.suffix == '.ps') ):
    # printing message
    print ("Output graphic file must be either EPS, PDF, PNG, or PS.")
    print ("Given graphic file name = %s" % file_graphic)
    # exit
    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
if (centroid == 'com'):
#    (x_centre, y_centre) = photutils.centroids.centroid_com (subframe_skysub)
    (x_centre, y_centre) = photutils.centroids.centroid_com (subframe)
elif (centroid == '1dg'):
#    (x_centre, y_centre) = photutils.centroids.centroid_1dg (subframe_skysub)
    (x_centre, y_centre) = photutils.centroids.centroid_1dg (subframe)
elif (centroid == '2dg'):
#    (x_centre, y_centre) = photutils.centroids.centroid_2dg (subframe_skysub)
    (x_centre, y_centre) = photutils.centroids.centroid_2dg (subframe)

# PSF fitting
subframe_y, subframe_x = numpy.indices (subframe_skysub.shape)
if (psf_model == '2dg'):
    psf_init = astropy.modeling.models.Gaussian2D (x_mean=x_centre, \
                                                   y_mean=y_centre)
elif (psf_model == '2dm'):
    psf_init = astropy.modeling.models.Moffat2D (x_0=x_centre, y_0=y_centre, \
                                                 amplitude=1.0, \
                                                 alpha=1.0, gamma=1.0)
fit = astropy.modeling.fitting.LevMarLSQFitter ()
```

```
psf_fitted = fit (psf_init, subframe_x, subframe_y, subframe_skysub, \
                  maxiter=maxiters)

# fitted PSF parameters
amplitude = psf_fitted.amplitude.value
if (psf_model == '2dg'):
    x_centre_sub = psf_fitted.x_mean.value
    y_centre_sub = psf_fitted.y_mean.value
    x_centre_psf = psf_fitted.x_mean.value + subframe_xmin
    y_centre_psf = psf_fitted.y_mean.value + subframe_ymin
    x_fwhm       = psf_fitted.x_fwhm
    y_fwhm       = psf_fitted.y_fwhm
    fwhm         = (x_fwhm + y_fwhm) / 2.0
    theta        = psf_fitted.theta.value
if (psf_model == '2dm'):
    x_centre_sub = psf_fitted.x_0.value
    y_centre_sub = psf_fitted.y_0.value
    x_centre_psf = psf_fitted.x_0.value + subframe_xmin
    y_centre_psf = psf_fitted.y_0.value + subframe_ymin
    alpha        = psf_fitted.alpha.value
    gamma        = psf_fitted.gamma.value
    fwhm         = psf_fitted.fwhm

# position of centre of star in pixel coordinate
position_pix = (x_centre_psf, y_centre_psf)

# 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)

# sky background estimate
sigma_clip = astropy.stats.SigmaClip (sigma=threshold, maxiters=maxiters)
apphot_sky_stats \
    = photutils.aperture.ApertureStats (data, apphot_annulus, \
                                        sigma_clip=sigma_clip)
skybg_per_pix     = apphot_sky_stats.mean
skybg_err_per_pix = apphot_sky_stats.std

# aperture photometry
phot_star    = photutils.aperture.aperture_photometry (data, apphot_aperture)
raw_flux     = phot_star['aperture_sum']
npix         = apphot_aperture.area
net_flux     = raw_flux - skybg_per_pix * npix
net_flux_err = numpy.sqrt (raw_flux + npix * skybg_err_per_pix**2)

# 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 ("#   (centroid_x, centroid_y)     = (%f, %f)\n" \
              % (x_centre + subframe_xmin, y_centre + subframe_ymin) )
    fh.write ("#   (centre_x, centre_y)         = (%f, %f)\n" \
              % (x_centre_psf, y_centre_psf) )
    fh.write ("#   FWHM of stellar PSF          = %f pixel\n" % fwhm)
    fh.write ("#   aperture radius              = %f pix\n" \
              % aperture_radius_pix)
    fh.write ("#   inner sky annulus            = %f pix\n" \
              % skyannulus_inner_pix)
    fh.write ("#   outer sky annulus            = %f pix\n" \
              % skyannulus_outer_pix)
    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, x_centre_psf, y_centre_psf, \
                 net_flux, net_flux_err, instmag, instmag_err, airmass) )

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

# axes
ax.set_xlabel ('RA')
```

```
ax.set_ylabel ('Dec')
ax.set_xlim (int (x_centre_psf - skyannulus_outer_pix * 1.2),
             int (x_centre_psf + skyannulus_outer_pix * 1.2) )
ax.set_ylim (int (y_centre_psf - skyannulus_outer_pix * 1.2),
             int (y_centre_psf + skyannulus_outer_pix * 1.2) )

# plotting image
norm \
    = astropy.visualization.mpl_normalize.ImageNormalize \
    ( stretch=astropy.visualization.HistEqStretch (data) )
im = ax.imshow (data, origin='lower', cmap=cmap, norm=norm)
fig.colorbar (im)

# making a circle to indicate the location of standard star
aperture = matplotlib.patches.Circle (xy=(x_centre_psf, y_centre_psf), \
                                      radius=aperture_radius_pix, \
                                      fill=False, color="red", linewidth=2)
annulus1 = matplotlib.patches.Circle (xy=(x_centre_psf, y_centre_psf), \
                                      radius=skyannulus_inner_pix, \
                                      fill=False, color="cyan", linewidth=2)
annulus2 = matplotlib.patches.Circle (xy=(x_centre_psf, y_centre_psf), \
                                      radius=skyannulus_outer_pix, \
                                      fill=False, color="cyan", linewidth=2)
# plotting location of standard star
ax.add_patch (aperture)
ax.add_patch (annulus1)
ax.add_patch (annulus2)

# invert Y-axis
ax.invert_yaxis ()

# saving file
fig.savefig (file_graphic, dpi=resolution)
```

Execute the script and check the results.

```
% chmod a+x advobs202202_s13_06.py
% ./advobs202202_s13_06.py -h
usage: advobs202202_s13_06.py [-h] [-i INPUT] [-o OUTPUT] [-g GRAPHIC]
                              [-c {com,1dg,2dg}] [-p {2dg,2dm}] [-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]
                              [-z {viridis,plasma,inferno,magma,cividis,binary,g
ray,bone,pink,spring,summer,autumn,winter,cool,hot,copper,ocean,terrain,gnuplot,
cubehelix,jet,turbo}]
                              [-l RESOLUTION]

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 data file name
```

```
  -g GRAPHIC, --graphic GRAPHIC
                        output graphic file name
  -c {com,1dg,2dg}, --centroid {com,1dg,2dg}
                        centroid measurement algorithm (default: 2dg)
  -p {2dg,2dm}, --psf {2dg,2dm}
                        PSF model [2dg=Gaussian, 2dm=Moffat] (default: 2dg)
  -r RA, --ra RA        RA in degree
  -d DEC, --dec DEC     Dec in degree
  -a APERTURE, --aperture APERTURE
                        aperture radius in FWHM (default: 2.0)
  -w HALFWIDTH, --halfwidth HALFWIDTH
                        half-width for centroid measurement (default: 20)
  -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: 4)
  -n MAXITERS, --maxiters MAXITERS
                        maximum number of iterations (default: 100)
  -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)
  -z {viridis,plasma,inferno,magma,cividis,binary,gray,bone,pink,spring,summer,a
utumn,winter,cool,hot,copper,ocean,terrain,gnuplot,cubehelix,jet,turbo}, --cmap
{viridis,plasma,inferno,magma,cividis,binary,gray,bone,pink,spring,summer,autumn
,winter,cool,hot,copper,ocean,terrain,gnuplot,cubehelix,jet,turbo}
                        choice of colour map (default: bone)
  -l RESOLUTION, --resolution RESOLUTION
                        resolution in DPI (default: 450)

% ./advobs202202_s13_06.py -r 162.59599 -d -0.0818 -c 2dg -p 2dg -z viridis \
? -i pg1047/lot_20210214_0145_df.fits -o test.phot -g test.pdf
WARNING: FITSFixedWarning: RADECSYS= 'FK5 ' / Equatorial coordinate system
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJD-OBS to 59259.725532
 from DATE-OBS'. [astropy.wcs.wcs]
% ls -lF test.*
-rw-r--r--  1 daisuke  taiwan  40608 Apr 28 14:03 test.pdf
-rw-r--r--  1 daisuke  taiwan   1641 Apr 28 14:03 test.phot
% cat test.phot
#
# Result of Aperture Photometry
#
#  Date/Time of Analysis
#   Date/Time = 2022-04-28T14:03:49.467309
#
#  Input Parameters
#   FITS file                   = pg1047/lot_20210214_0145_df.fits
#   RA                          = 162.595990 deg
#   Dec                         = -0.081800 deg
#   aperture radius             = 2.000000 in FWHM
#   inner sky annulus           = 4.000000 in FWHM
#   outer sky annulus           = 7.000000 in FWHM
#   half-width for centroid     = 20.000000 pixel
#   threshold for sigma-clipping = 4.000000 in sigma
```

```
#   number of max iterations     = 100
#   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)
#   (centroid_x, centroid_y)     = (559.972936, 1553.999991)
#   (centre_x, centre_y)         = (559.973017, 1553.999895)
#   FWHM of stellar PSF          = 4.472550 pixel
#   aperture radius              = 8.945101 pix
#   inner sky annulus            = 17.890202 pix
#   outer sky annulus            = 31.307853 pix
#   sky background level         = 9.425092 ADU per pixel
#   net flux                     = 55173.494737 ADU
#   net flux err                 = 277.999524 ADU
#   instrumental mag             = -9.354326
#   instrumental mag err         = 0.005471
#
#  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 559.973017 1553.9998
95 55173.494737 277.999524 -9.354326 0.005471 1.094255
```

Check the size of aperture and sky annulus. (Fig. 2)

```
% okular test.pdf
```

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: advobs202202_s13_07.py

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

#
# Time-stamp: <2022/04/28 14:23:54 (CST) daisuke>
#

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing subprocess module
import subprocess

# importing astropy module
import astropy.io.fits

# constructing parser object
```

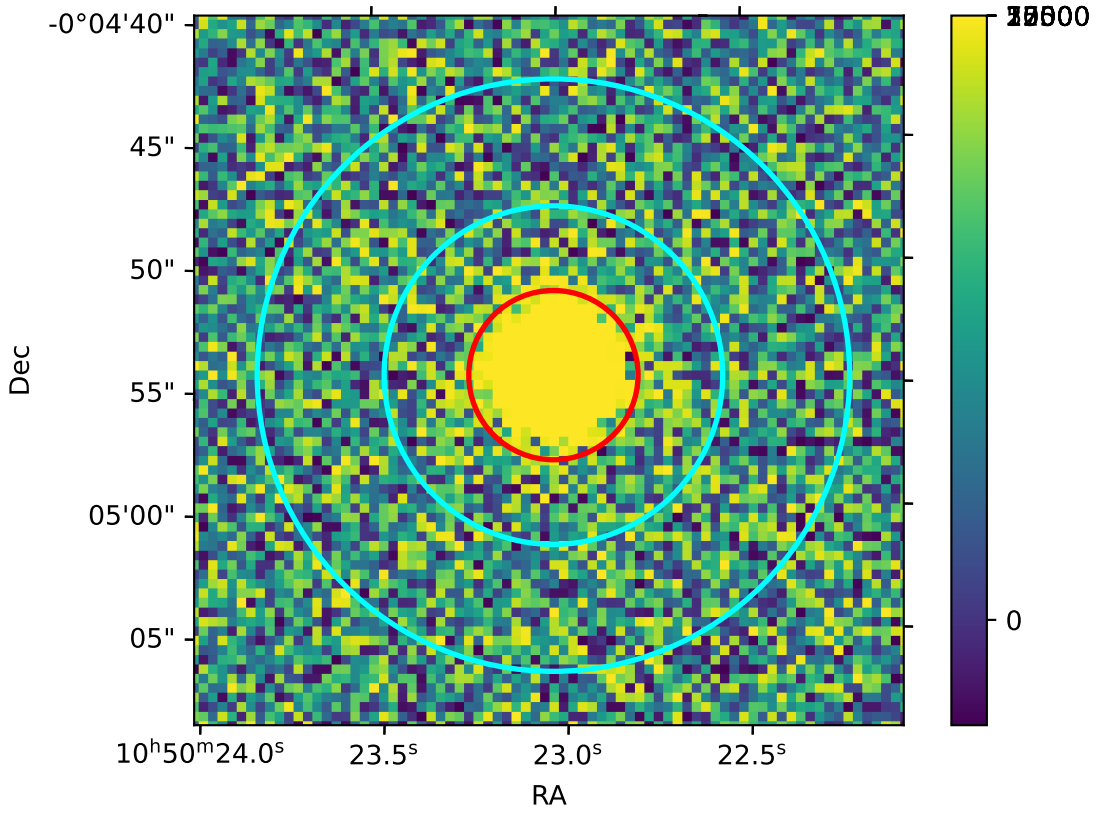

Figure 2: The aperture and sky annulus for the photometry.

```
desc   = "carrying out aperture photometry for multiple FITS files"
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=2.0, \
                     help='aperture radius in FWHM (default: 2.0)')
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 ('-o', '--object', default='', \
                     help='target object name')
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 == ''):
    # printing message
    print ("You have to specify target object name by using -n option!")
    # exit
    sys.exit ()
if (filter_name == ''):
    # printing message
    print ("You have to specify filter name by using -f option!")
    # exit
    sys.exit ()

# processing each FITS file
for file_fits in files_fits:
    # making pathlib object
    path_fits = pathlib.Path (file_fits)
    # file name
    filename = path_fits.name
    name_list = file_fits.split ('_')
    frame_id = int (name_list[-2])

    # output data file name
    file_output  = "phot_%s_%s_%04d.phot" % (target_name, filter_name, frame_id)
    # output graphic file name
    file_graphic = "phot_%s_%s_%04d.pdf" % (target_name, filter_name, frame_id)
```

```
    # if the file is not a FITS file, then skip
    if not (path_fits.suffix == '.fits'):
        # skip
        continue
    # if the file is not a reduced data, then skip
    if not (path_fits.stem[-3:] == '_df'):
        # skip
        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['FILTER'] == filter_name) \
             and (header['EXPTIME'] > exptime_min) \
             and (header['EXPTIME'] < exptime_max) ):
        continue

    # command to carry out photometry
    command_phot \
        = ("./advobs202202_s13_06.py -i %s -r %f -d %f -a %f -o %s -g %s" \
           % (file_fits, target_ra_deg, target_dec_deg, \
              aperture_radius_fwhm, file_output, file_graphic) )

    # execute the command
    subprocess.run (command_phot, shell=True)
```

Run the script and carry out photometry for all the g'-band data.

```
% chmod a+x advobs202202_s13_07.py
% ./advobs202202_s13_07.py -h
usage: advobs202202_s13_07.py [-h] [-f FILTER] [-n NAME] [-e1 EXPTIME_MIN]
                              [-e2 EXPTIME_MAX] [-a APERTURE] [-r RA] [-d DEC]
                              [-o OBJECT]
                              files [files ...]

carrying out aperture photometry for multiple FITS files

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: 2.0)
  -r RA, --ra RA        RA in degree
  -d DEC, --dec DEC     Dec in degree
  -o OBJECT, --object OBJECT
                        target object name
```

```
% ./advobs202202_s13_07.py -r 162.59599 -d -0.0818 -f gp_Astrodon_2019 \
? -n id08 pg1047/*.fits
% ls phot_id08_gp_Astrodon_2019_0*
phot_id08_gp_Astrodon_2019_0145.pdf     phot_id08_gp_Astrodon_2019_0286.phot
phot_id08_gp_Astrodon_2019_0145.phot    phot_id08_gp_Astrodon_2019_0287.pdf
phot_id08_gp_Astrodon_2019_0146.pdf     phot_id08_gp_Astrodon_2019_0287.phot
phot_id08_gp_Astrodon_2019_0146.phot    phot_id08_gp_Astrodon_2019_0325.pdf
phot_id08_gp_Astrodon_2019_0147.pdf     phot_id08_gp_Astrodon_2019_0325.phot
phot_id08_gp_Astrodon_2019_0147.phot    phot_id08_gp_Astrodon_2019_0326.pdf
phot_id08_gp_Astrodon_2019_0245.pdf     phot_id08_gp_Astrodon_2019_0326.phot
phot_id08_gp_Astrodon_2019_0245.phot    phot_id08_gp_Astrodon_2019_0327.pdf
phot_id08_gp_Astrodon_2019_0246.pdf     phot_id08_gp_Astrodon_2019_0327.phot
phot_id08_gp_Astrodon_2019_0246.phot    phot_id08_gp_Astrodon_2019_0366.pdf
phot_id08_gp_Astrodon_2019_0247.pdf     phot_id08_gp_Astrodon_2019_0366.phot
phot_id08_gp_Astrodon_2019_0247.phot    phot_id08_gp_Astrodon_2019_0367.pdf
phot_id08_gp_Astrodon_2019_0285.pdf     phot_id08_gp_Astrodon_2019_0367.phot
phot_id08_gp_Astrodon_2019_0285.phot    phot_id08_gp_Astrodon_2019_0368.pdf
phot_id08_gp_Astrodon_2019_0286.pdf     phot_id08_gp_Astrodon_2019_0368.phot
```

Photometric measurements were done for 16 FITS files.

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: advobs202202_s13_08.py

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

#
# Time-stamp: <2022/04/28 14:52:13 (CST) daisuke>
#

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# 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 ('-o', '--output', default='', help='output image file')
```

```
parser.add_argument ('-t', '--title', default='', help='title of the plot')
parser.add_argument ('-r', '--resolution', type=int, default=450, \
                     help='resolution of output graphic file (default: 450)')
parser.add_argument ('files', nargs='+', help='input data files')

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

# input parameters
file_output = args.output
list_input  = args.files
resolution  = args.resolution
title       = args.title

# making pathlib object
path_output = pathlib.Path (file_output)

# checking file_output
if (file_output == ''):
    # printing message
    print ("Output file name must be given.")
    # exit
    sys.exit ()
if not ( (path_output.suffix == '.eps') or (path_output.suffix == '.pdf') \
         or (path_output.suffix == '.png') or (path_output.suffix == '.ps') ):
    # printing message
    print ("Output image file must be either EPS, PDF, PNG, or PS.")
    print ("Given output image file name = %s" % file_output)
    # exit
    sys.exit ()

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

# processing each input data file
for file_input in list_input:
    # making pathlib object
    path_input = pathlib.Path (file_input)
    # existence check
    if not (path_input.exists ()):
        # printing message
        print ("### data file '%s' does not exist, skipping..." % file_input)
        # skip
        continue

    # opening input data file
    with open (file_input, 'r') as fh:
        # reading file line-by-line
        for line in fh:
            # if the line starts with '#', then skip
            if (line[0] == '#'):
                # skip
                continue
            # 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

# text for extinction coefficient
text_coeff = "extinction coefficient: %5.3f +/- %5.3f mag/airmass" \
    % (a_fitted, a_err)

# 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, 'm--', label='least-squares fitting')
ax.errorbar (array_airmass, array_instmag, yerr=array_instmag_err, \
             fmt='go', ecolor='black', capsize=5, \
             label='photometric standard star')
ax.text (0.05, 0.05, text_coeff, transform=ax.transAxes)
ax.set_title (title)
ax.legend ()

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

Execute the script, and determine g'-band atmospheric extinction coefficient. (Fig. 3)

```
% chmod a+x advobs202202_s13_08.py
% ./advobs202202_s13_08.py -h
usage: advobs202202_s13_08.py [-h] [-o OUTPUT] [-t TITLE] [-r RESOLUTION]
                              files [files ...]

determining atmospheric extinction coefficient

positional arguments:
  files                 input data files

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        output image file
  -t TITLE, --title TITLE
                        title of the plot
  -r RESOLUTION, --resolution RESOLUTION
                        resolution of output graphic file (default: 450)

% ./advobs202202_s13_08.py -t "g'-band atmospheric extinction, 14/Feb/2021" \
? -o extinction_g.pdf phot_id08_gp_Astrodon_2019_0*.phot
popt:
[ 0.12098169 -9.47834611]
pcov:
[[ 5.44409906e-05 -7.79730609e-05]
 [-7.79730609e-05  1.14173423e-04]]
dof = 13
reduced chi^2 = 6.454710303069199e-05
a = 0.120982 +/- 0.007378 (6.098786 %)
b = -9.478346 +/- 0.010685 (-0.112733 %)
% ls -lF extinction_g.pdf
-rw-r--r--  1 daisuke  taiwan  18289 Apr 28 14:56 extinction_g.pdf
```

The g'-band atmospheric extinction coefficient is estimated to be 0.121 ± 0.007 mag/airmass.
Display the plot. (Fig. 3)

```
% okular extinction_g.pdf
```

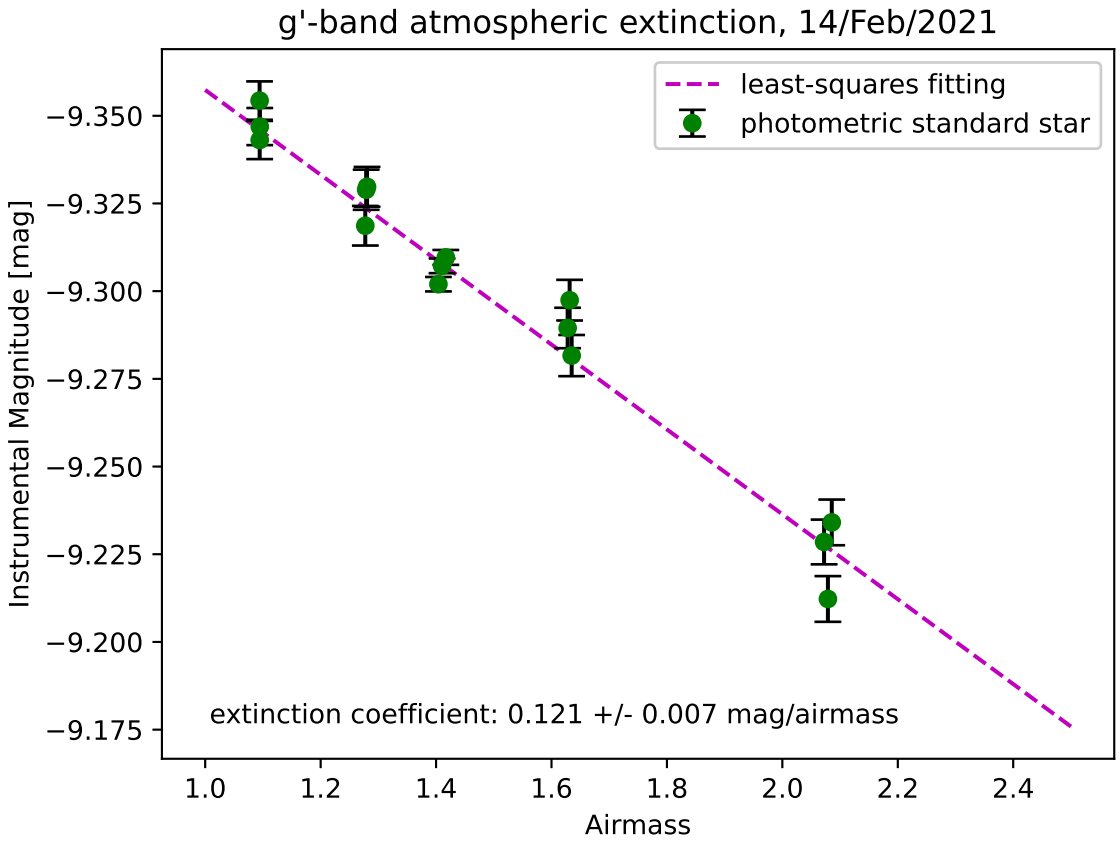

Figure 3: Atmospheric extinction at SDSS g'-band. The atmospheric extinction coefficient at g'-band is 0.121 ± 0.007 mag/airmass.

4 Processing r'-band data

4.1 Photometry of the star ID 8 on r'-band data

Carry out aperture photometry of the star ID 8 on r'-band data.

```
% ./advobs202202_s13_07.py -r 162.59599 -d -0.0818 -f rp_Astrodon_2019 \
? -n id08 pg1047/*.fits
% ls phot_id08_rp_Astrodon_2019_0*
phot_id08_rp_Astrodon_2019_0148.pdf     phot_id08_rp_Astrodon_2019_0289.phot
phot_id08_rp_Astrodon_2019_0148.phot    phot_id08_rp_Astrodon_2019_0290.pdf
phot_id08_rp_Astrodon_2019_0149.pdf     phot_id08_rp_Astrodon_2019_0290.phot
phot_id08_rp_Astrodon_2019_0149.phot    phot_id08_rp_Astrodon_2019_0328.pdf
phot_id08_rp_Astrodon_2019_0150.pdf     phot_id08_rp_Astrodon_2019_0328.phot
phot_id08_rp_Astrodon_2019_0150.phot    phot_id08_rp_Astrodon_2019_0329.pdf
phot_id08_rp_Astrodon_2019_0248.pdf     phot_id08_rp_Astrodon_2019_0329.phot
phot_id08_rp_Astrodon_2019_0248.phot    phot_id08_rp_Astrodon_2019_0330.pdf
phot_id08_rp_Astrodon_2019_0249.pdf     phot_id08_rp_Astrodon_2019_0330.phot
phot_id08_rp_Astrodon_2019_0249.phot    phot_id08_rp_Astrodon_2019_0369.pdf
phot_id08_rp_Astrodon_2019_0250.pdf     phot_id08_rp_Astrodon_2019_0369.phot
```

```
phot_id08_rp_Astrodon_2019_0250.phot    phot_id08_rp_Astrodon_2019_0370.pdf
phot_id08_rp_Astrodon_2019_0288.pdf     phot_id08_rp_Astrodon_2019_0370.phot
phot_id08_rp_Astrodon_2019_0288.phot    phot_id08_rp_Astrodon_2019_0371.pdf
phot_id08_rp_Astrodon_2019_0289.pdf     phot_id08_rp_Astrodon_2019_0371.phot
```

4.2 Atmospheric extinction coefficient in r'-band

Use least-square method to estimate atmospheric extinction coefficient in r'-band.

```
% ./advobs202202_s13_08.py -t "r'-band atmospheric extinction, 14/Feb/2021" \
? -o extinction_r.pdf phot_id08_rp_Astrodon_2019_0*.phot
popt:
[  0.07311288 -10.15490352]
pcov:
[[ 4.19062121e-05 -6.10697240e-05]
 [-6.10697240e-05  9.12656362e-05]]
dof = 13
reduced chi^2 = 5.3622070571868915e-05
a = 0.073113 +/- 0.006474 (8.854118 %)
b = -10.154904 +/- 0.009553 (-0.094076 %)
% ls -l extinction_*.pdf
-rw-r--r--  1 daisuke  taiwan  18289 Apr 28 14:56 extinction_g.pdf
-rw-r--r--  1 daisuke  taiwan  17901 Apr 28 15:10 extinction_r.pdf
```

The estimated atmospheric extinction coefficient in r'-band is 0.073 ± 0.006 mag/airmass.
Display the plot. (Fig. 4)

```
% okular extinction_r.pdf
```

5 Processing i'-band data

5.1 Photometry of the star ID 8 on i'-band data

Carry out aperture photometry of the star ID 8 on i'-band data.

```
% ./advobs202202_s13_07.py -r 162.59599 -d -0.0818 -f ip_Astrodon_2019 \
? -n id08 pg1047/*.fits
% ls phot_id08_ip_Astrodon_2019_0*
phot_id08_ip_Astrodon_2019_0151.pdf     phot_id08_ip_Astrodon_2019_0292.phot
phot_id08_ip_Astrodon_2019_0151.phot    phot_id08_ip_Astrodon_2019_0293.pdf
phot_id08_ip_Astrodon_2019_0152.pdf     phot_id08_ip_Astrodon_2019_0293.phot
phot_id08_ip_Astrodon_2019_0152.phot    phot_id08_ip_Astrodon_2019_0331.pdf
phot_id08_ip_Astrodon_2019_0153.pdf     phot_id08_ip_Astrodon_2019_0331.phot
phot_id08_ip_Astrodon_2019_0153.phot    phot_id08_ip_Astrodon_2019_0332.pdf
phot_id08_ip_Astrodon_2019_0251.pdf     phot_id08_ip_Astrodon_2019_0332.phot
phot_id08_ip_Astrodon_2019_0251.phot    phot_id08_ip_Astrodon_2019_0333.pdf
phot_id08_ip_Astrodon_2019_0252.pdf     phot_id08_ip_Astrodon_2019_0333.phot
phot_id08_ip_Astrodon_2019_0252.phot    phot_id08_ip_Astrodon_2019_0372.pdf
phot_id08_ip_Astrodon_2019_0253.pdf     phot_id08_ip_Astrodon_2019_0372.phot
phot_id08_ip_Astrodon_2019_0253.phot    phot_id08_ip_Astrodon_2019_0373.pdf
phot_id08_ip_Astrodon_2019_0291.pdf     phot_id08_ip_Astrodon_2019_0373.phot
phot_id08_ip_Astrodon_2019_0291.phot    phot_id08_ip_Astrodon_2019_0374.pdf
phot_id08_ip_Astrodon_2019_0292.pdf     phot_id08_ip_Astrodon_2019_0374.phot
```

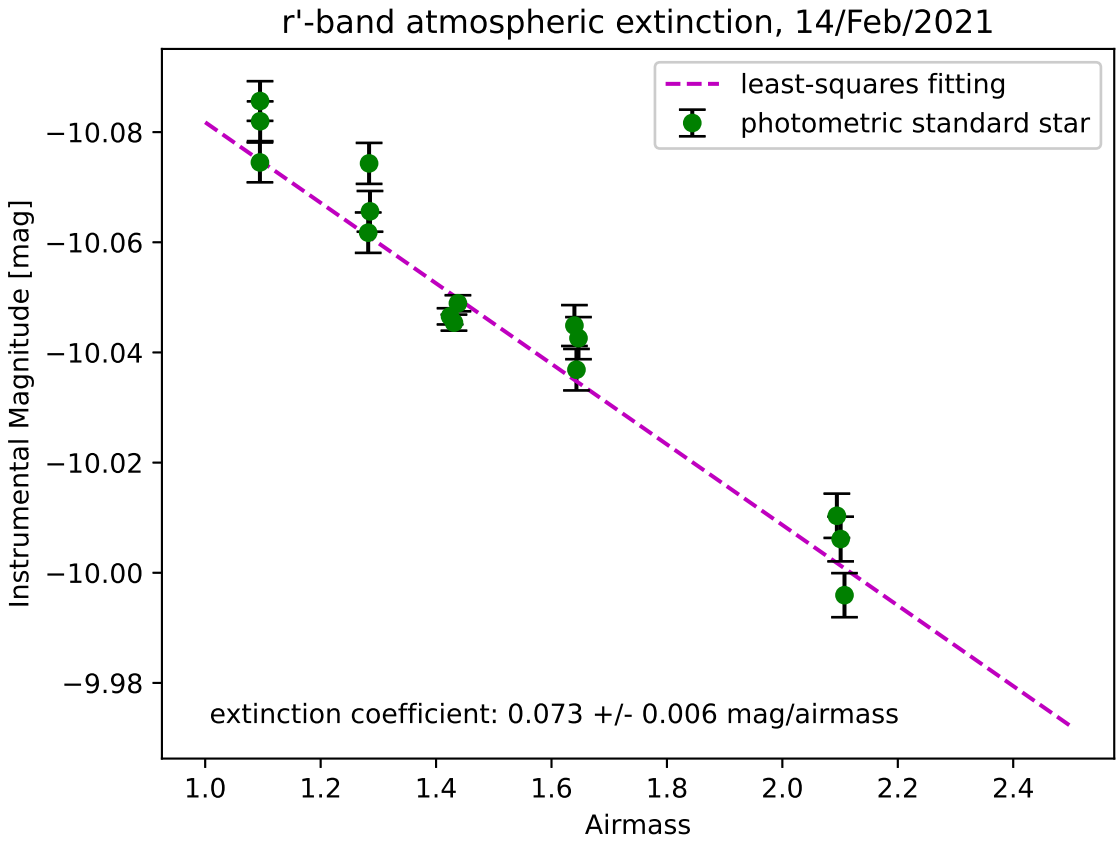

Figure 4: Atmospheric extinction at SDSS r'-band. The atmospheric extinction coefficient at r'-band is 0.073 ± 0.006 mag/airmass.

5.2 Atmospheric extinction coefficient in r'-band

Use least-square method to estimate atmospheric extinction coefficient in i'-band.

```
% ./advobs202202_s13_08.py -t "i'-band atmospheric extinction, 14/Feb/2021" \
? -o extinction_i.pdf phot_id08_ip_Astrodon_2019_0*.phot
popt:
[ 0.02534082 -9.6433137 ]
pcov:
[[ 2.99762658e-05 -4.42969492e-05]
 [-4.42969492e-05  6.71484896e-05]]
dof = 13
reduced chi^2 = 5.3779370634188414e-05
a = 0.025341 +/- 0.005475 (21.605686 %)
b = -9.643314 +/- 0.008194 (-0.084975 %)
% ls -l extinction_*.pdf
-rw-r--r--  1 daisuke  taiwan  18289 Apr 28 14:56 extinction_g.pdf
-rw-r--r--  1 daisuke  taiwan  18080 Apr 28 15:14 extinction_i.pdf
-rw-r--r--  1 daisuke  taiwan  17901 Apr 28 15:10 extinction_r.pdf
```

The estimated atmospheric extinction coefficient in i'-band is 0.025 ± 0.005 mag/airmass.
Display the plot. (Fig. 5)

```
% okular extinction_i.pdf
```

6 For your further reading

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`

7 Exercise

1. What is airmass? Describe about it. How can we calculate airmass? How important to consider airmass for observation and data analysis?
2. Atmospheric extinction:
   (a) What is atmospheric extinction? Describe about it. How to determine atmospheric extinction coefficients? How important to consider atmospheric extinction for observation and data analysis?
   (b) Atmospheric extinction coefficients
      i. What is typical value of U-band atmospheric extinction coefficient? Show the references you have gathered.
      ii. What is typical value of B-band atmospheric extinction coefficient? Show the references you have gathered.

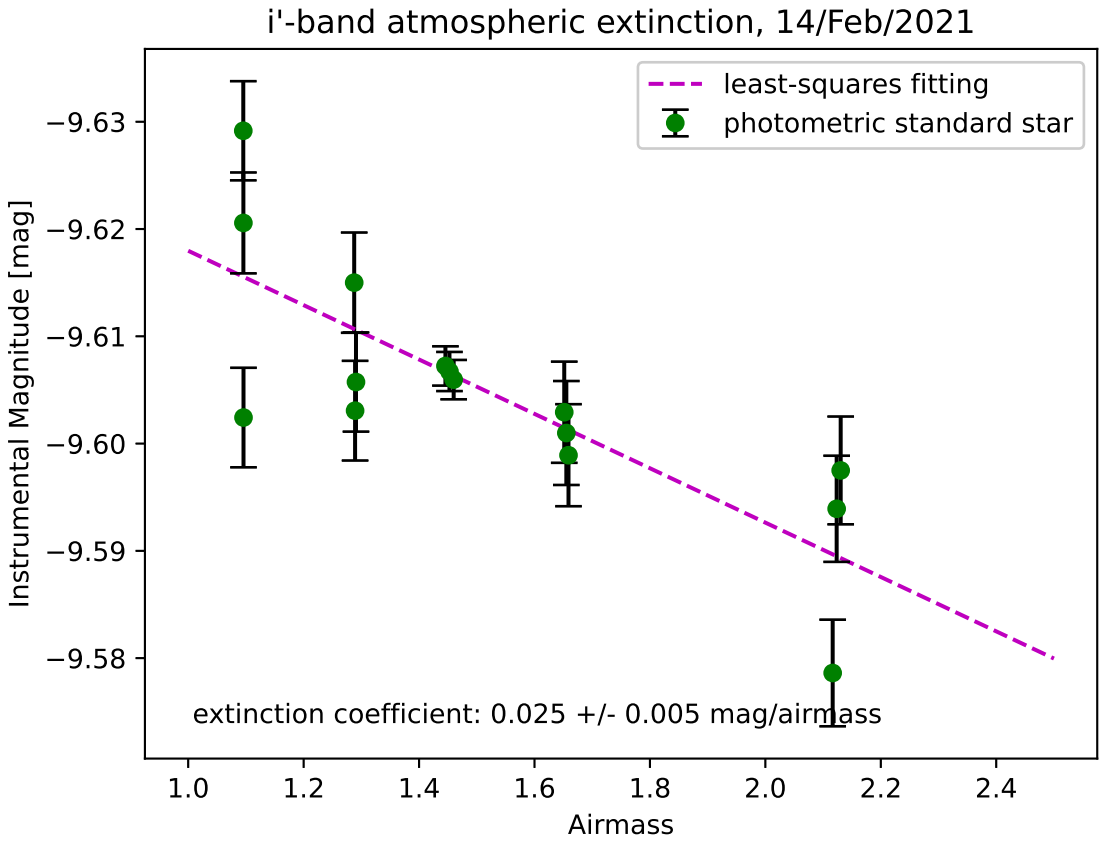

Figure 5: Atmospheric extinction at SDSS i'-band. The atmospheric extinction coefficient at r'-band is 0.025 ± 0.005 mag/airmass.

iii. What is typical value of V-band atmospheric extinction coefficient? Show the references you have gathered.
iv. What is typical value of R-band atmospheric extinction coefficient? Show the references you have gathered.
v. What is typical value of I-band atmospheric extinction coefficient? Show the references you have gathered.
vi. What is typical value of u'-band atmospheric extinction coefficient? Show the references you have gathered.
vii. What is typical value of g'-band atmospheric extinction coefficient? Show the references you have gathered.
viii. What is typical value of r'-band atmospheric extinction coefficient? Show the references you have gathered.
ix. What is typical value of i'-band atmospheric extinction coefficient? Show the references you have gathered.
x. What is typical value of z'-band atmospheric extinction coefficient? Show the references you have gathered.

3. Choose a star other than the ID 8 in PG1047+003 field. Estimate g'-, r'-, and i'-band atmospheric extinction coefficient. Show the results of your analysis. Show the source code of your Python scripts used for this analysis.

4. Compare values of g'-band and r'-band atmospheric extinction coefficients. Is g'-band atmospheric extinction coefficient tends to be larger than r'-band atmospheric extinction coefficient? Or, Is g'-band atmospheric extinction coefficient tends to be smaller than r'-band atmospheric extinction coefficient? Or, Is g'-band atmospheric extinction coefficient tends to be roughly the same with r'-band atmospheric extinction coefficient? Why? What is the background physics determining the value of atmospheric extinction coefficient?

5. Download publicly available archived data. Describe the data you have downloaded. Try to measure the atmospheric extinction coefficient. Show the results of your analysis. Show the source code of your Python scripts.