Advanced Astronomical Observations 2021
Session 16: Sky Background Brightness

Kinoshita Daisuke

19 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 measure sky background brightness.

1 Python scripts for this session

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

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

```
% git clone https://github.com/kinoshitadaisuke/ncu_advobs_202102.git
Cloning into 'ncu_advobs_202102'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 23 (delta 4), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (23/23), 23.39 KiB | 23.39 MiB/s, done.
Resolving deltas: 100% (4/4), done.
% ls ncu_advobs_202102
LICENSE             README.md           s16_skybackground/
% ls ncu_advobs_202102/s16_skybackground/
README              ao2021_s16_03.py    ao2021_s16_06.py    ao2021_s16_09.py
ao2021_s16_01.py    ao2021_s16_04.py    ao2021_s16_07.py
ao2021_s16_02.py    ao2021_s16_05.py    ao2021_s16_08.py
% head -15 ncu_advobs_202102/s16_skybackground/ao2021_s16_01.py
#!/usr/pkg/bin/python3.9

# importing argparse module
```

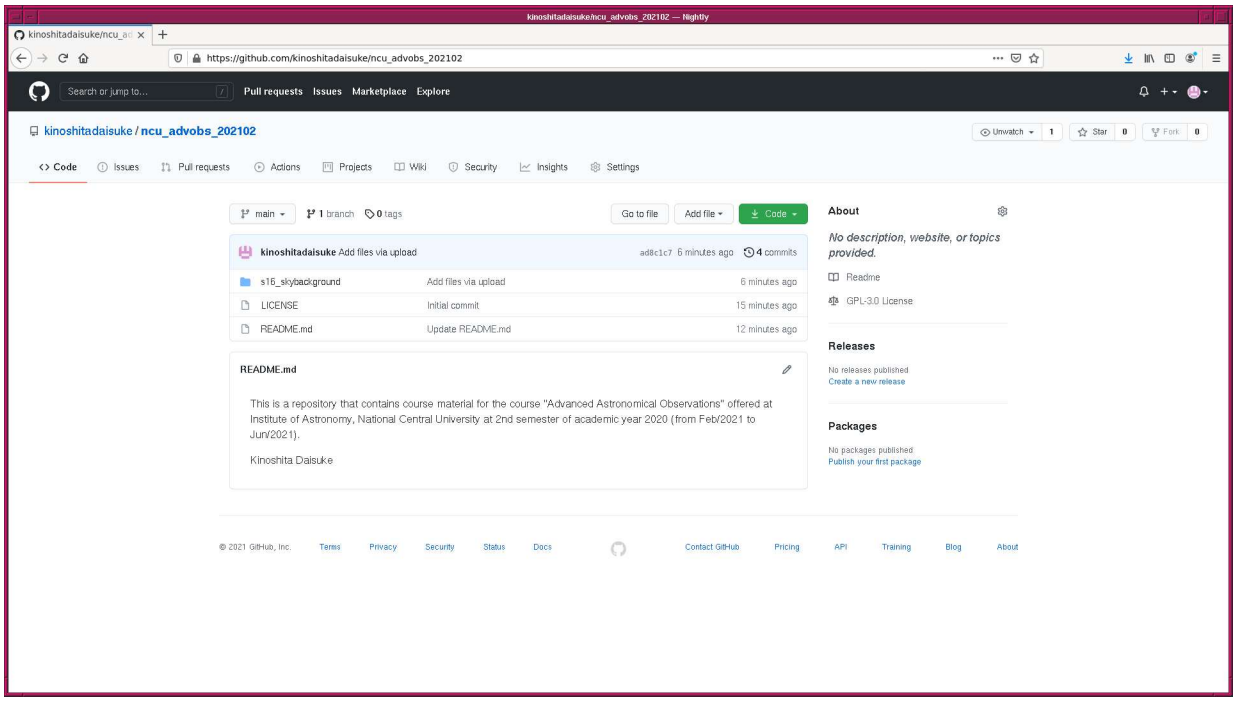

Figure 1: The GitHub repository for Python scripts for this course.

```
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.io.fits

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

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

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

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

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

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

2 Selecting and copying FITS files to process

2.1 Selecting FITS files to process

We measure the sky background brightness. The data should be (1) an image of a photometric standard star field, and (2) an image of relatively longer exposure. Search for longer exposure data of a photometric standard star field from the data taken on 14 February 2021.

Make a Python script to select FITS files satisfying given criteria.

Python Code 1: ao2021_s16_01.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.io.fits

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

# adding arguments
parser.add_argument ('-n', '--name', default='', help='target object name')
parser.add_argument ('-e1', '--exptime-min', type=float, default=0.0, \
                     help='minimum exposure time in sec')
parser.add_argument ('-e2', '--exptime-max', type=float, default=3600.0, \
                     help='maximum exposure time in sec')
parser.add_argument ('-a', '--keyword-airmass', default='AIRMASS', \
                     help='FITS keyword for airmass (default: AIRMASS)')
parser.add_argument ('-d', '--keyword-datatype', default='IMAGETYP', \
                     help='FITS keyword for data type (default: IMAGETYP)')
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 ('-o', '--keyword-object', default='OBJECT', \
                     help='FITS keyword for object name (default: OBJECT)')
parser.add_argument ('-t', '--keyword-timeobs', default='TIME-OBS', \
                     help='FITS keyword for TIME-OBS (default: TIME-OBS)')
parser.add_argument ('files', nargs='+', help='FITS files')

# parsing arguments
args = parser.parse_args ()

# input parameters
target_name      = args.name
exptime_min      = args.exptime_min
exptime_max      = args.exptime_max
keyword_airmass  = args.keyword_airmass
keyword_datatype = args.keyword_datatype
keyword_exptime  = args.keyword_exptime
keyword_filter   = args.keyword_filter
keyword_object   = args.keyword_object
keyword_timeobs  = args.keyword_timeobs
files_fits       = args.files

# check of target object name
if (target_name == ''):
    print ("Target object name must be specified.")
    sys.exit ()

# making an empty dictionary for storing search results
```

```
dic_fits = {}

# processing each FITS file
for file_fits in files_fits:
    # check of file name
    if not (file_fits[-5:] == '.fits'):
        print ("The file \"%s\" is not a FITS file!" % file_fits)
        print ("File must be a FITS file. Skipping.")
        continue

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

    # if target object name does not match with specified target object name,
    # then skip
    if not (header[keyword_object] == target_name):
        continue

    # if exposure time does not match with specified exposure time criteria,
    # then skip
    if not ( (header[keyword_exptime] >= exptime_min) \
             and (header[keyword_exptime] <= exptime_max) ):
        continue

    # adding information to the dictionary
    dic_fits[file_fits] = {}
    dic_fits[file_fits][keyword_airmass]  = header[keyword_airmass]
    dic_fits[file_fits][keyword_datatype] = header[keyword_datatype]
    dic_fits[file_fits][keyword_exptime]  = header[keyword_exptime]
    dic_fits[file_fits][keyword_filter]   = header[keyword_filter]
    dic_fits[file_fits][keyword_object]   = header[keyword_object]
    dic_fits[file_fits][keyword_timeobs]  = header[keyword_timeobs]

# printing results
for file_fits in dic_fits.keys ():
    print (file_fits)
    print ("  %-8s = %-24s    %-8s = %-24s" \
           % (keyword_datatype, dic_fits[file_fits][keyword_datatype], \
              keyword_object, dic_fits[file_fits][keyword_object]) )
    print ("  %-8s = %-24s    %-8s = %f sec" \
           % (keyword_filter, dic_fits[file_fits][keyword_filter], \
              keyword_exptime, dic_fits[file_fits][keyword_exptime]) )
    print ("  %-8s = %-24s    %-8s = %f" \
           % (keyword_timeobs, dic_fits[file_fits][keyword_timeobs], \
              keyword_airmass, dic_fits[file_fits][keyword_airmass]) )
```

Execute the script, and search for FITS files that we look for.

```
% chmod a+x ao2021_s16_01.py
% ./ao2021_s16_01.py -h
usage: ao2021_s16_01.py [-h] [-n NAME] [-e1 EXPTIME_MIN] [-e2 EXPTIME_MAX]
                        [-a KEYWORD_AIRMASS] [-d KEYWORD_DATATYPE]
                        [-e KEYWORD_EXPTIME] [-f KEYWORD_FILTER]
                        [-o KEYWORD_OBJECT] [-t KEYWORD_TIMEOBS]
                        files [files ...]
```

```
selecting FITS files

positional arguments:
  files                 FITS files

optional arguments:
  -h, --help            show this help message and exit
  -n NAME, --name NAME  target object name
  -e1 EXPTIME_MIN, --exptime-min EXPTIME_MIN
                        minimum exposure time in sec
  -e2 EXPTIME_MAX, --exptime-max EXPTIME_MAX
                        maximum exposure time in sec
  -a KEYWORD_AIRMASS, --keyword-airmass KEYWORD_AIRMASS
                        FITS keyword for airmass (default: AIRMASS)
  -d KEYWORD_DATATYPE, --keyword-datatype KEYWORD_DATATYPE
                        FITS keyword for data type (default: IMAGETYP)
  -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)
  -o KEYWORD_OBJECT, --keyword-object KEYWORD_OBJECT
                        FITS keyword for object name (default: OBJECT)
  -t KEYWORD_TIMEOBS, --keyword-timeobs KEYWORD_TIMEOBS
                        FITS keyword for TIME-OBS (default: TIME-OBS)

% ./ao2021_s16_01.py -n PG1047 -e1 120 -e2 300 \
? ../../14_reduction3/script_14/ccdred_20210511_122633/*_df.fits
../../14_reduction3/script_14/ccdred_20210511_122633/lot_20210214_0185_df.fits
  IMAGETYP = LIGHT                      OBJECT   = PG1047
  FILTER   = gp_Astrodon_2019           EXPTIME  = 180.000000 sec
  TIME-OBS = 17:58:41                   AIRMASS  = 1.122154
../../14_reduction3/script_14/ccdred_20210511_122633/lot_20210214_0186_df.fits
  IMAGETYP = LIGHT                      OBJECT   = PG1047
  FILTER   = gp_Astrodon_2019           EXPTIME  = 180.000000 sec
  TIME-OBS = 18:01:59                   AIRMASS  = 1.126160
../../14_reduction3/script_14/ccdred_20210511_122633/lot_20210214_0187_df.fits
  IMAGETYP = LIGHT                      OBJECT   = PG1047
  FILTER   = rp_Astrodon_2019           EXPTIME  = 180.000000 sec
  TIME-OBS = 18:05:25                   AIRMASS  = 1.130614
../../14_reduction3/script_14/ccdred_20210511_122633/lot_20210214_0188_df.fits
  IMAGETYP = LIGHT                      OBJECT   = PG1047
  FILTER   = rp_Astrodon_2019           EXPTIME  = 180.000000 sec
  TIME-OBS = 18:08:41                   AIRMASS  = 1.135143
../../14_reduction3/script_14/ccdred_20210511_122633/lot_20210214_0189_df.fits
  IMAGETYP = LIGHT                      OBJECT   = PG1047
  FILTER   = ip_Astrodon_2019           EXPTIME  = 180.000000 sec
  TIME-OBS = 18:12:06                   AIRMASS  = 1.140164
../../14_reduction3/script_14/ccdred_20210511_122633/lot_20210214_0190_df.fits
  IMAGETYP = LIGHT                      OBJECT   = PG1047
  FILTER   = ip_Astrodon_2019           EXPTIME  = 180.000000 sec
  TIME-OBS = 18:15:23                   AIRMASS  = 1.145258
```

We have found six FITS files.

2.2 Copying FITS files to process

Make a Python script to copy FITS files that we have selected.

Python Code 2: ao2021_s16_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   = "selecting FITS files"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-n', '--name', default='', help='target object name')
parser.add_argument ('-e1', '--exptime-min', type=float, default=0.0, \
                     help='minimum exposure time in sec')
parser.add_argument ('-e2', '--exptime-max', type=float, default=3600.0, \
                     help='maximum exposure time in sec')
parser.add_argument ('-a', '--keyword-airmass', default='AIRMASS', \
                     help='FITS keyword for airmass (default: AIRMASS)')
parser.add_argument ('-d', '--keyword-datatype', default='IMAGETYP', \
                     help='FITS keyword for data type (default: IMAGETYP)')
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 ('-o', '--keyword-object', default='OBJECT', \
                     help='FITS keyword for object name (default: OBJECT)')
parser.add_argument ('-t', '--keyword-timeobs', default='TIME-OBS', \
                     help='FITS keyword for TIME-OBS (default: TIME-OBS)')
parser.add_argument ('files', nargs='+', help='FITS files')

# parsing arguments
args = parser.parse_args ()

# input parameters
target_name      = args.name
exptime_min      = args.exptime_min
exptime_max      = args.exptime_max
keyword_airmass  = args.keyword_airmass
keyword_datatype = args.keyword_datatype
keyword_exptime  = args.keyword_exptime
keyword_filter   = args.keyword_filter
keyword_object   = args.keyword_object
keyword_timeobs  = args.keyword_timeobs
files_fits       = args.files

# check of target object name
if (target_name == ''):
```

```
    print ("Target object name must be specified.")
    sys.exit ()

# making an empty dictionary for storing search results
dic_fits = {}

# processing each FITS file
for file_fits in files_fits:
    # check of file name
    if not (file_fits[-5:] == '.fits'):
        print ("The file \"%s\" is not a FITS file!" % file_fits)
        print ("File must be a FITS file. Skipping.")
        continue

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

    # if target object name does not match with specified target object name,
    # then skip
    if not (header[keyword_object] == target_name):
        continue

    # if exposure time does not match with specified exposure time criteria,
    # then skip
    if not ( (header[keyword_exptime] >= exptime_min) \
             and (header[keyword_exptime] <= exptime_max) ):
        continue

    # adding information to the dictionary
    dic_fits[file_fits] = {}
    dic_fits[file_fits][keyword_airmass]  = header[keyword_airmass]
    dic_fits[file_fits][keyword_datatype] = header[keyword_datatype]
    dic_fits[file_fits][keyword_exptime]  = header[keyword_exptime]
    dic_fits[file_fits][keyword_filter]   = header[keyword_filter]
    dic_fits[file_fits][keyword_object]   = header[keyword_object]
    dic_fits[file_fits][keyword_timeobs]  = header[keyword_timeobs]

# copying files
for file_fits in dic_fits.keys ():
    # making pathlib object
    path_fits = pathlib.Path (file_fits)
    # if the file exists, then copy the file to currently working directory
    if (path_fits.exists () ):
        # printing status
        print ("Processing the file \"%s\"..." % path_fits.name)
        print ("  now copying the file \"%s\"..." % path_fits.name)
        # copying file
        shutil.copy2 (path_fits, '.')
        print ("  finished copying the file \"%s\"!" % path_fits.name)
```

Run the script, and copy files to the currently working directory.

```
% chmod a+x ao2021_s16_02.py
% ./ao2021_s16_02.py -h
usage: ao2021_s16_02.py [-h] [-n NAME] [-e1 EXPTIME_MIN] [-e2 EXPTIME_MAX]
                        [-a KEYWORD_AIRMASS] [-d KEYWORD_DATATYPE]
```

```
                        [-e KEYWORD_EXPTIME] [-f KEYWORD_FILTER]
                        [-o KEYWORD_OBJECT] [-t KEYWORD_TIMEOBS]
                        files [files ...]

selecting FITS files

positional arguments:
  files                 FITS files

optional arguments:
  -h, --help            show this help message and exit
  -n NAME, --name NAME  target object name
  -e1 EXPTIME_MIN, --exptime-min EXPTIME_MIN
                        minimum exposure time in sec
  -e2 EXPTIME_MAX, --exptime-max EXPTIME_MAX
                        maximum exposure time in sec
  -a KEYWORD_AIRMASS, --keyword-airmass KEYWORD_AIRMASS
                        FITS keyword for airmass (default: AIRMASS)
  -d KEYWORD_DATATYPE, --keyword-datatype KEYWORD_DATATYPE
                        FITS keyword for data type (default: IMAGETYP)
  -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)
  -o KEYWORD_OBJECT, --keyword-object KEYWORD_OBJECT
                        FITS keyword for object name (default: OBJECT)
  -t KEYWORD_TIMEOBS, --keyword-timeobs KEYWORD_TIMEOBS
                        FITS keyword for TIME-OBS (default: TIME-OBS)

% ls *.fits
ls: No match.
% ./ao2021_s16_02.py -n PG1047 -e1 120 -e2 300 \
? ../../14_reduction3/script_14/ccdred_20210511_122633/*_df.fits
Processing the file "lot_20210214_0185_df.fits"...
  now copying the file "lot_20210214_0185_df.fits"...
  finished copying the file "lot_20210214_0185_df.fits"!
Processing the file "lot_20210214_0186_df.fits"...
  now copying the file "lot_20210214_0186_df.fits"...
  finished copying the file "lot_20210214_0186_df.fits"!
Processing the file "lot_20210214_0187_df.fits"...
  now copying the file "lot_20210214_0187_df.fits"...
  finished copying the file "lot_20210214_0187_df.fits"!
Processing the file "lot_20210214_0188_df.fits"...
  now copying the file "lot_20210214_0188_df.fits"...
  finished copying the file "lot_20210214_0188_df.fits"!
Processing the file "lot_20210214_0189_df.fits"...
  now copying the file "lot_20210214_0189_df.fits"...
  finished copying the file "lot_20210214_0189_df.fits"!
Processing the file "lot_20210214_0190_df.fits"...
  now copying the file "lot_20210214_0190_df.fits"...
  finished copying the file "lot_20210214_0190_df.fits"!
% ls *.fits
lot_20210214_0185_df.fits               lot_20210214_0188_df.fits
lot_20210214_0186_df.fits               lot_20210214_0189_df.fits
lot_20210214_0187_df.fits               lot_20210214_0190_df.fits
```

2.3 Visual inspection of the image

We first process the file `lot_20210214_0185_df.fits`. Use Ginga to show the image of tt lot_20210214_0185_df.fits. (Fig. 2)

```
% ginga lot_20210214_0185_df.fits &
```

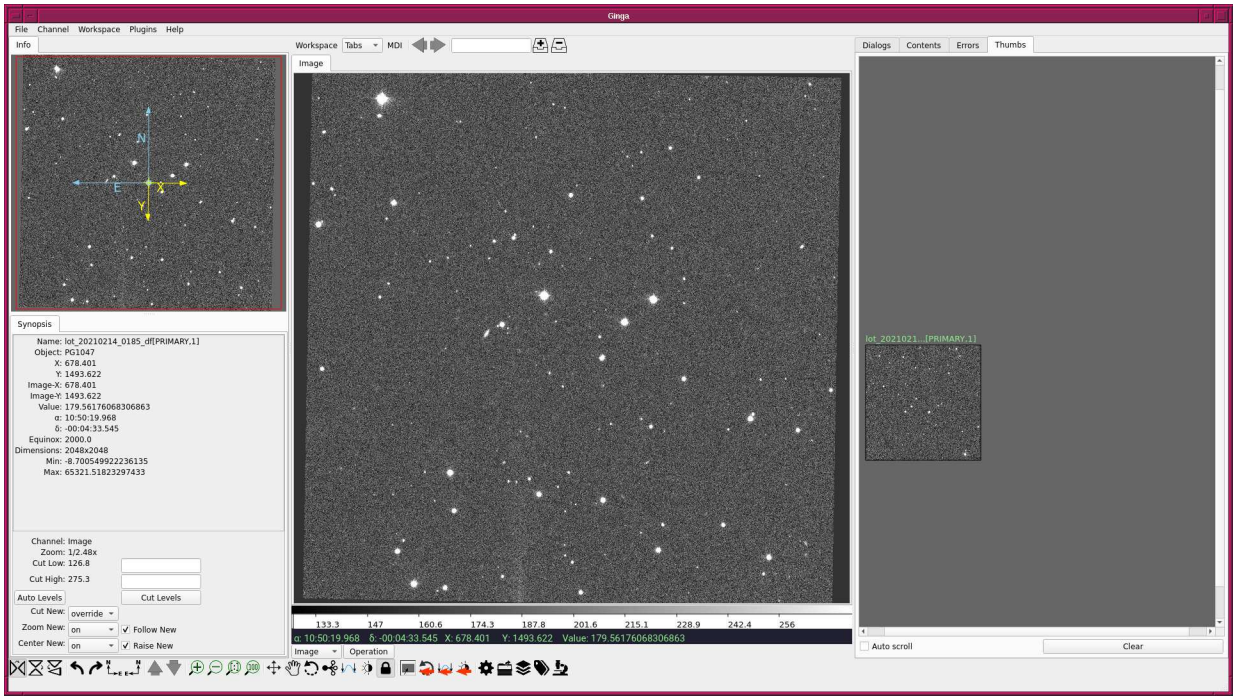

Figure 2: The image of `lot_20210214_0185_df.fits` displayed using Ginga.

3 Photometric standard stars

3.1 Downloading photometric standard star catalogue

Download the photometric standard star catalogue.

```
% 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  21265      0  0:00:03  0:00:03 --:--:-- 21265
% ls -l stds.tar.gz
-rw-r--r--  1 daisuke  taiwan  65774 May 18 20:53 stds.tar.gz
```

3.2 Extracting photometric standard star catalogue

Extract the photometric standard star catalogue.

```
% mkdir stds
% ls
ao2021_s16_01.py*                    lot_20210214_0187_df.fits
ao2021_s16_01.py~                    lot_20210214_0188_df.fits
```

```
ao2021_s16_02.py*                       lot_20210214_0189_df.fits
ao2021_s16_02.py~                       lot_20210214_0190_df.fits
lot_20210214_0185_df.fits               stds/
lot_20210214_0186_df.fits               stds.tar.gz
% 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 ..
```

3.3 Showing coordinates of photometric standards in PG 1047+003 field

Make a Python script to show the coordinates of photometric standard stars in the field PG 1047+003.

Python Code 3: ao2021_s16_03.py

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

# importing argparse module
import argparse

# importing astropy module
import astropy.coordinates

# constructing parser object
desc   = "showing coordinates of photometric standard stars"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('files', nargs='+', help='photometric standard star files')

# parsing arguments
args = parser.parse_args ()
```

```
# input parameters
files_cat = args.files

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

# processing each catalogue file
for file_cat in files_cat:
    # opening the file
    with open (file_cat, '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, and show coordinates of stars in the field of PG 1047+003.

```
% chmod a+x ao2021_s16_03.py
% ./ao2021_s16_03.py -h
usage: ao2021_s16_03.py [-h] files [files ...]

showing coordinates of photometric standard stars

positional arguments:
  files       photometric standard star files

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

% ./ao2021_s16_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
```

3.4 Marking photometric standard stars

Make a Python script to mark photometric standard stars using Photutils package.

Python Code 4: ao2021_s16_04.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 objects'
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 ('-c', '--catalogue', default='', \
                     help='standard star catalogue file name')
parser.add_argument ('-r', '--radius', type=float, default=10.0, \
                     help='radius of aperture in arcsec (default: 10)')
parser.add_argument ('-a', '--z1', type=float, default=0.0, \
                     help='minimum value to display (default: 0)')
parser.add_argument ('-b', '--z2', type=float, default=100.0, \
                     help='maximum value to display (default: 100)')

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

```
# input parameters
file_fits      = args.input
file_output    = args.output
file_cat       = args.catalogue
radius_arcsec  = args.radius
z1             = args.z1
z2             = args.z2

# unit
u_arcsec = astropy.units.arcsec

# 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 ()

# check of catalogue file
if not (file_cat[-13:] == '.txt.clean.v3'):
    print ("Catalogue file must be SDSS photometric standard star catalogue.")
    print ("Given catalogue file name = %s" % file_cat)
    sys.exit ()

# making an empty dictionary to store data for standards
dic_stds = {}

# opening catalogue file
with open (file_cat, 'r') as fh:
    # reading the file line-by-line
    for line in fh:
        # if the line stars 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])
        # adding data to the dictionary
        if not (star_id in dic_stds):
            dic_stds[star_id] = {}
            dic_stds[star_id]['RA_deg'] = ra_deg
            dic_stds[star_id]['Dec_deg'] = dec_deg

# positions of standard stars
positions = []
for star in dic_stds.keys ():
    positions.append ( (dic_stds[star]['RA_deg'], dic_stds[star]['Dec_deg']) )
coords = astropy.coordinates.SkyCoord (positions, 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 apertures
apertures_sky \
    = photutils.aperture.SkyCircularAperture (coords, \
                                              r=radius_arcsec * u_arcsec)
apertures_pix = apertures_sky.to_pixel (wcs)

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

Run the script, and make a PDF file. Show the PDF file using the command `xpdf`. (Fig. 3)

```
% chmod a+x ao2021_s16_04.py
% ./ao2021_s16_04.py -h
usage: ao2021_s16_04.py [-h] [-i INPUT] [-o OUTPUT] [-c CATALOGUE] [-r RADIUS]
                        [-a Z1] [-b Z2]

marking target objects

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
  -c CATALOGUE, --catalogue CATALOGUE
                        standard star catalogue file name
  -r RADIUS, --radius RADIUS
                        radius of aperture in arcsec (default: 10)
  -a Z1, --z1 Z1        minimum value to display (default: 0)
  -b Z2, --z2 Z2        maximum value to display (default: 100)

% ./ao2021_s16_04.py -i lot_20210214_0185_df.fits -o pg1047.pdf \
? -c stds/PG1047+003A.txt.clean.v3 -r 20 -a 0 -b 500
WARNING: FITSFixedWarning: RADECSYS= 'FK5 ' / Equatorial coordinate system
the RADECSYS keyword is deprecated, use RADESYSa. [astropy.wcs.wcs]
% ls -l pg1047.pdf
-rw-r--r--  1 daisuke  taiwan  688020 May 18 21:31 pg1047.pdf
% xpdf pg1047.pdf
```

4 Selecting a standard star

Check peak counts of standard stars, and select a star for the measurement.
Use Ginga to display the image.

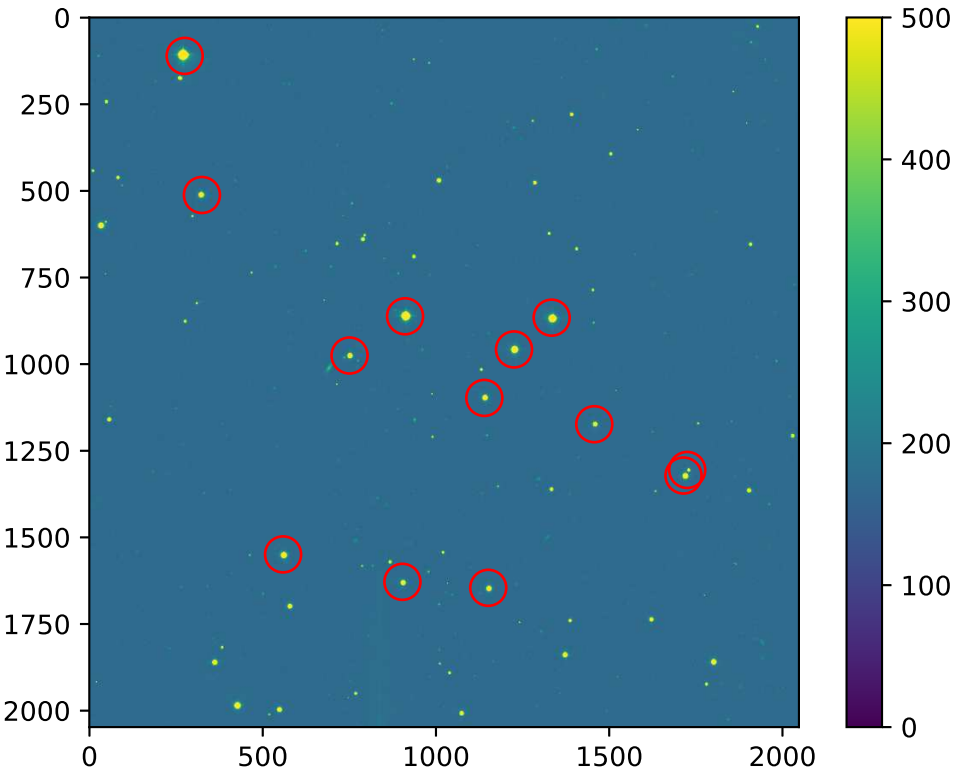

Figure 3: Photometric standard stars in the field of PG 1047+003.

```
% ginga lot_20210214_0185_df.fits &
```

Click the menu ‘Plugins‘, move your mouse cursor to ‘Analysis‘, then choose ‘Pick‘. Move your mouse cursor to the position of a photometric standard star, and click the left-button of your mouse. Show the radial plot. If the peak count is $\sim$ 65,000 ADU, the image of the star is saturated. (Fig. 4) Find a photometric standard star with the peak count of $\sim$ 10,000 – 20,000 ADU. (Fig. 5) For following analysis, we use the star #025.

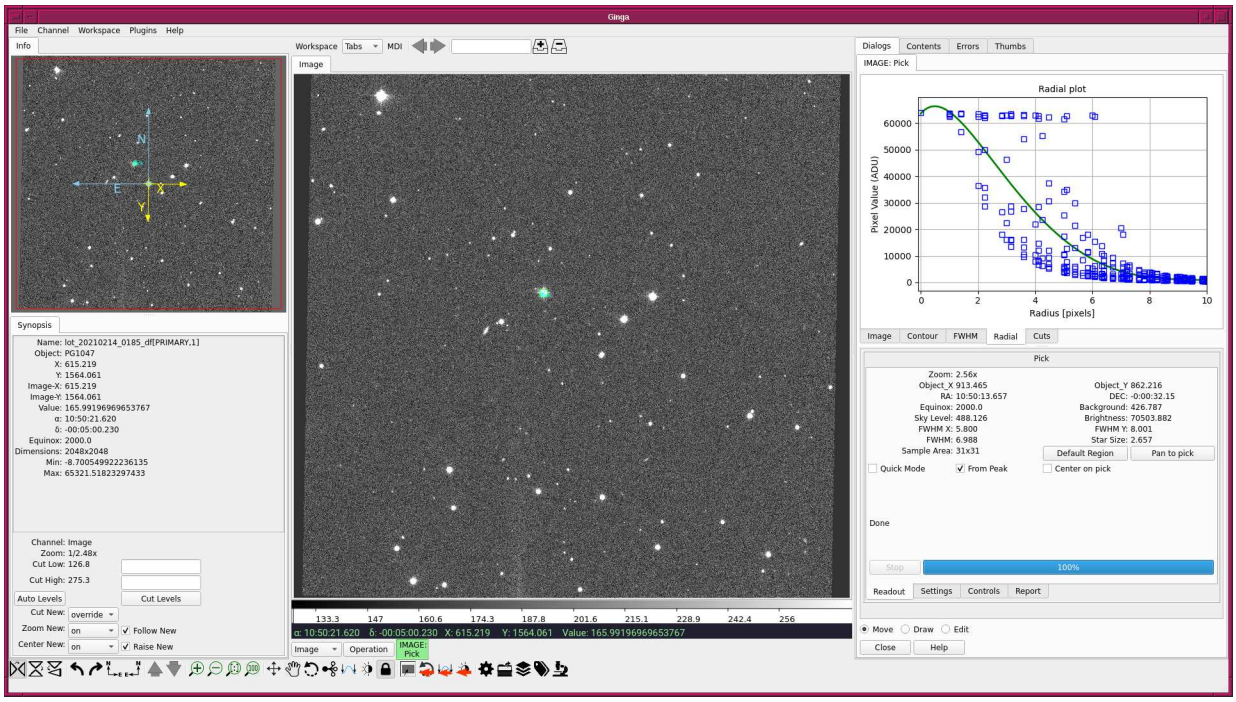

Figure 4: An example of radial plot of a saturated star.

5 Aperture photometry of a standard star

Make a Python script to carry out aperture photometry of a selected photometric standard star.

Python Code 5: ao2021_s16_05.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
```

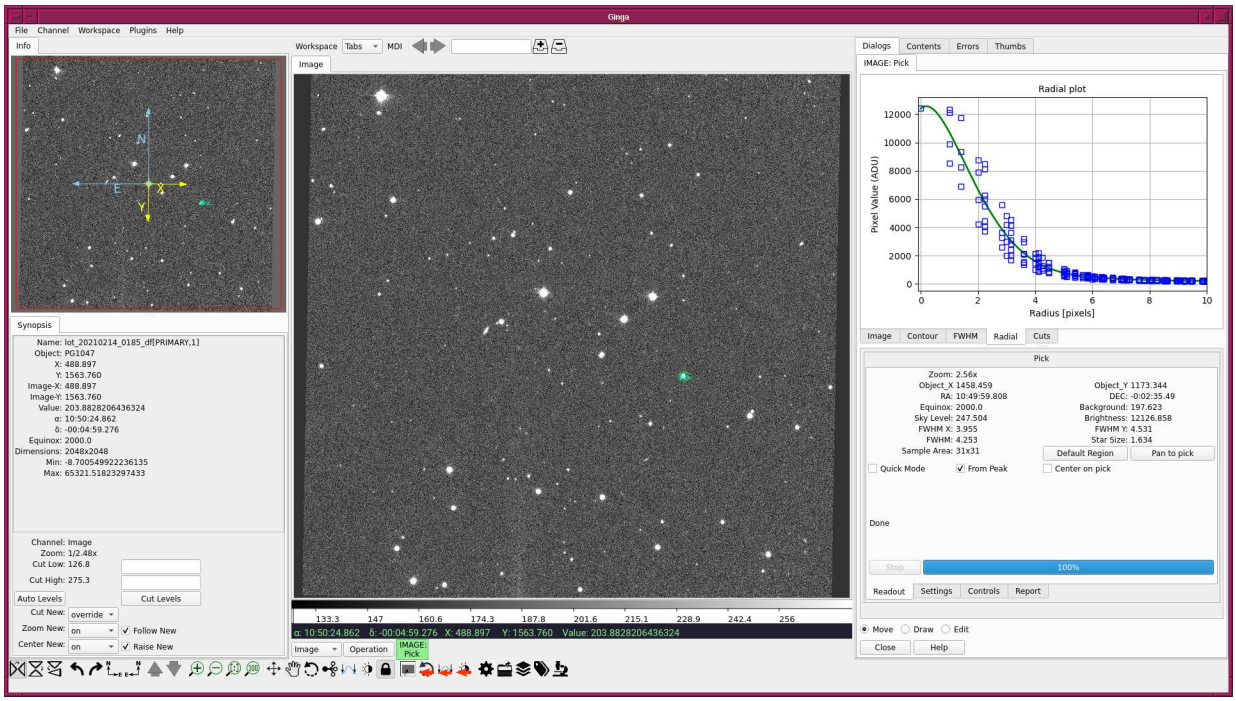

Figure 5: An example of radial plot of a non-saturated star.

```
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, \
                     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 carry out aperture photometry of the star #025.

```
% chmod a+x ao2021_s16_05.py
% ./ao2021_s16_05.py -h
usage: ao2021_s16_05.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_s16_05.py -i lot_20210214_0185_df.fits -o phot_0185_025.data \
? -r 162.49926 -d -0.0433
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/16_skybackground/script_16/./ao
2021_s16_05.py:179: RuntimeWarning: invalid value encountered in sqrt
  noise         = numpy.sqrt (data)
% ls -l phot_0185_025.data
-rw-r--r--  1 daisuke  taiwan  1417 May 18 22:08 phot_0185_025.data
% cat phot_0185_025.data
#
# Result of Aperture Photometry
#
#  Date/Time of Analysis
#   Date/Time = 2021-05-18T22:08:03.584497
```

```
.....

#  Results
#   file, exptime, filter, centre_x, centre_y,
#   net_flux, net_flux_err, instmag, instmag_err, airmass
lot_20210214_0185_df.fits 180.000000 gp_Astrodon_2019 1457.429651 1172.355437 28
1649.633641 551.994273 -7.986092 0.002128 1.122154
```

6 Measuring sky background level

6.1 Constructing a histogram of pixel values

Make a Python script to construct a histogram of pixel values of given FITS file.

Python Code 6: ao2021_s16_06.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.io.fits

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

# constructing parser object
desc   = "making a histogram"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-i', '--input', default='', help='input FITS file')
parser.add_argument ('-o', '--output', default='', help='output file')
parser.add_argument ('-a', '--z1', type=float, default=0.0, \
                     help='minimum pixel value (default: 0)')
parser.add_argument ('-b', '--z2', type=float, default=70000.0, \
                     help='maximum pixel value (default: 70000)')
parser.add_argument ('-n', '--nbins', type=int, default=7000, \
                     help='number of bins (default: 7000)')

# parsing arguments
args = parser.parse_args ()

# input parameters
file_input  = args.input
file_output = args.output
z1          = args.z1
z2          = args.z2
nbins       = args.nbins

# check of input FITS file
if not (file_input[-5:] == '.fits'):
```

```
    print ("Input file must be a FITS file.")
    print ("Input file name = %s" % file_input)
    sys.exit ()

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

# opening FITS file
with astropy.io.fits.open (file_input) as hdu_list:
    # reading image data
    data = hdu_list[0].data

# flattening data
data_flat = data.flatten ()

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

# axes
ax.set_xlabel ("Pixel Value [ADU]")
ax.set_ylabel ("Number of pixels")

# plotting image
ax.set_xlim (z1, z2)
ax.hist (data_flat, bins=nbins, range=(z1, z2), histtype='bar', align='mid', \
         label=file_input)
ax.legend ()

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

Run the script, and generate a histogram. (Fig. 6) The mean background level seems to be $\sim$ 175 ADU.

```
% chmod a+x ao2021_s16_06.py
% ./ao2021_s16_06.py -h
usage: ao2021_s16_06.py [-h] [-i INPUT] [-o OUTPUT] [-a Z1] [-b Z2] [-n NBINS]

making a histogram

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input FITS file
  -o OUTPUT, --output OUTPUT
                        output file
  -a Z1, --z1 Z1        minimum pixel value (default: 0)
  -b Z2, --z2 Z2        maximum pixel value (default: 70000)
  -n NBINS, --nbins NBINS
                        number of bins (default: 7000)

% ./ao2021_s16_06.py -i lot_20210214_0185_df.fits -o 0185_hist.pdf \
? -a=0 -b=300 -n 3000
```

```
% ls -l 0185_hist.pdf
-rw-r--r--  1 daisuke  taiwan  61719 May 18 22:52 0185_hist.pdf
% xpdf 0185_hist.pdf
```

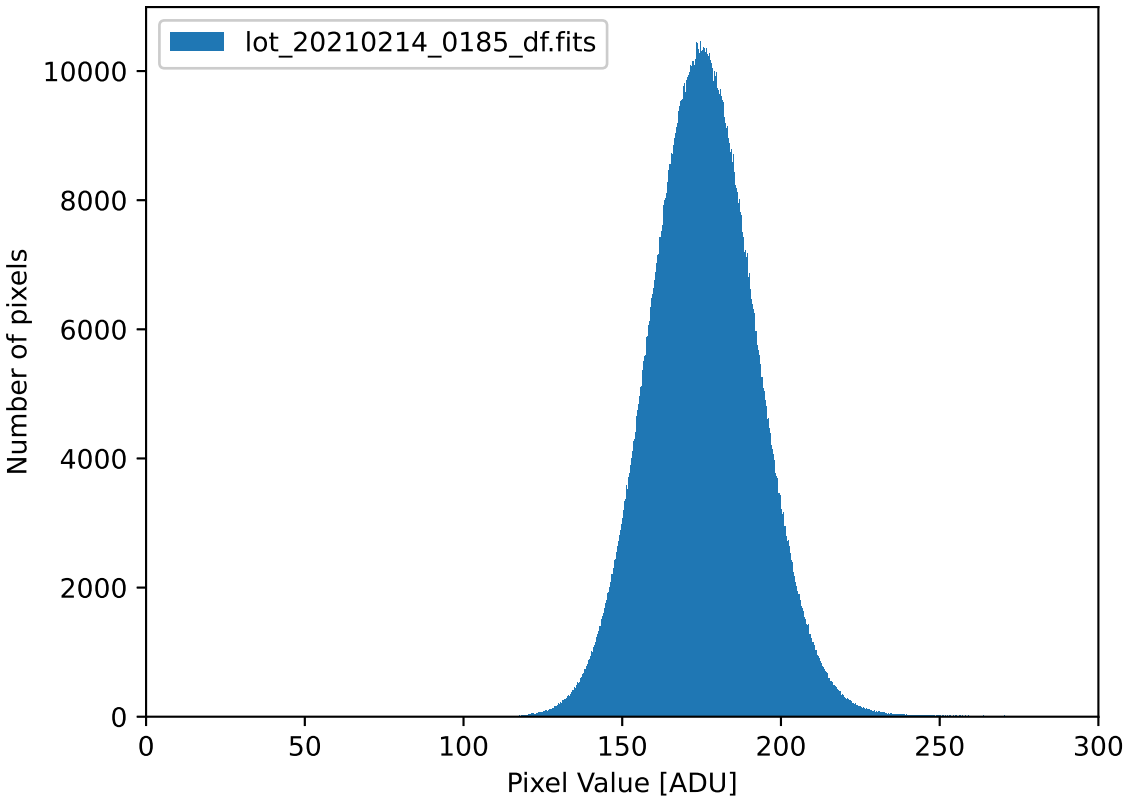

Figure 6: The histogram of the image `lot_20210214_0185_df.fits`.

6.2 Estimation of mode of distribution using empirical formula

Make a Python script to estimate mode of a given distribution using following empirical formula.

$$mode = 3 \times median - 2 \times mean \tag{1}$$

Python Code 7: ao2021_s16_07.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.io.fits
import astropy.stats
```

```
# constructing parser object
desc   = "estimating mode"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-i', '--input', default='', help='input FITS file')
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='number of maximum iterations (default: 10)')

# parsing arguments
args = parser.parse_args ()

# input parameters
file_input = args.input
threshold  = args.threshold
maxiters   = args.maxiters

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

# opening FITS file
with astropy.io.fits.open (file_input) as hdu_list:
    # reading image data
    data = hdu_list[0].data

# calculation of mean and median using sigma-clipping
mean, median, stddev \
    = astropy.stats.sigma_clipped_stats (data, sigma=threshold, \
                                         maxiters=maxiters, cenfunc='median')

# calculation of mode using empirical formula
mode = 3 * median - 2 * mean

print ("mean                          = %10.3f ADU" % mean)
print ("median                        = %10.3f ADU" % median)
print ("mode = 3 * median - 2 * mean = %10.3f ADU" % mode)
print ("stddev                        = %10.3f ADU" % stddev)
```

Execute the script.

```
% chmod a+x ao2021_s16_07.py
% ./ao2021_s16_07.py -h
usage: ao2021_s16_07.py [-h] [-i INPUT] [-t THRESHOLD] [-n MAXITERS]

estimating mode

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input FITS file
  -t THRESHOLD, --threshold THRESHOLD
                        threshold for sigma-clipping in sigma (default: 3)
  -n MAXITERS, --maxiters MAXITERS
```

```
                        number of maximum iterations (default: 10)

% ./ao2021_s16_07.py -i lot_20210214_0185_df.fits -t 4
mean                          =    175.566 ADU
median                        =    175.348 ADU
mode = 3 * median - 2 * mean  =    174.911 ADU
stddev                        =     16.414 ADU
```

The sky background level is estimated to be 175 ± 16 ADU.

7 Calculating pixel scale

Make a Python script to calculate pixel scale of the image.

Python Code 8: ao2021_s16_08.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# constructing parser object
desc   = "calculating pixel scale"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-i', '--input', default='', help='input FITS file name')
parser.add_argument ('-x', '--keyword-pixsize-x', default='XPIXSZ', \
                     help='FITS keyword for pixel width in micron')
parser.add_argument ('-y', '--keyword-pixsize-y', default='YPIXSZ', \
                     help='FITS keyword for pixel height in micron')
parser.add_argument ('-l', '--keyword-focallength', default='FOCALLEN', \
                     help='FITS keyword for focal length in mm')

# parsing arguments
args = parser.parse_args ()

# input parameters
file_input          = args.input
keyword_pixelsize_x = args.keyword_pixsize_x
keyword_pixelsize_y = args.keyword_pixsize_y
keyword_focallength = args.keyword_focallength

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

```
# opening FITS file
with astropy.io.fits.open (file_input) as hdu_list:
    # reading header information
    header = hdu_list[0].header

# pixel size and focal length
pixelsize_x_micron = header[keyword_pixelsize_x]
pixelsize_y_micron = header[keyword_pixelsize_y]
focallength_mm     = header[keyword_focallength]

# calculation of pixel scale
pixelscale_x_rad = pixelsize_x_micron * 10**-6 / (focallength_mm * 10**-3)
pixelscale_y_rad = pixelsize_y_micron * 10**-6 / (focallength_mm * 10**-3)
pixelscale_x_arcsec = pixelscale_x_rad * 180.0 / numpy.pi * 3600.0
pixelscale_y_arcsec = pixelscale_y_rad * 180.0 / numpy.pi * 3600.0

# printing result
print ("pixel scale on x-axis = %6.4f arcsec/pix" % pixelscale_x_arcsec)
print ("pixel scale on y-axis = %6.4f arcsec/pix" % pixelscale_y_arcsec)
```

Run the script, and show the result of pixel scale calculations.

```
% chmod a+x ao2021_s16_08.py
% ./ao2021_s16_08.py -h
usage: ao2021_s16_08.py [-h] [-i INPUT] [-x KEYWORD_PIXSIZE_X]
                        [-y KEYWORD_PIXSIZE_Y] [-l KEYWORD_FOCALLENGTH]

calculating pixel scale

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input FITS file name
  -x KEYWORD_PIXSIZE_X, --keyword-pixsize-x KEYWORD_PIXSIZE_X
                        FITS keyword for pixel width in micron
  -y KEYWORD_PIXSIZE_Y, --keyword-pixsize-y KEYWORD_PIXSIZE_Y
                        FITS keyword for pixel height in micron
  -l KEYWORD_FOCALLENGTH, --keyword-focallength KEYWORD_FOCALLENGTH
                        FITS keyword for focal length in mm

% ./ao2021_s16_08.py -i lot_20210214_0185_df.fits
pixel scale on x-axis = 0.3855 arcsec/pix
pixel scale on y-axis = 0.3855 arcsec/pix
```

The pixel scale is 0.3855 arcsec/pix. Therefore, the area of a pixel of the image projected on the sky is 0.1486 arcsec^2.

```
% python3.9
Python 3.9.2 (default, May  2 2021, 01:43:22)
[GCC 7.5.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> ps = 0.3855
>>> ps**2
0.14861025
>>> exit ()
```

8 Calculating sky background brightness in mag/arcsec^2

Make a Python script to calculate the sky background brightness in mag/arcsec^2.

Python Code 9: ao2021_s16_09.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing numpy module
import numpy

# constructing parser object
desc   = "calculating sky background brightness"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-b', '--skybg', type=float, default=-999.99, \
                     help='sky background level in ADU')
parser.add_argument ('-e', '--skybg-err', type=float, default=-999.99, \
                     help='error of sky background level in ADU')
parser.add_argument ('-s', '--star', type=float, default=-999.99, \
                     help='net flux of standard star in ADU')
parser.add_argument ('-m', '--mag', type=float, default=-999.99, \
                     help='apparent magnitude of standard star')
parser.add_argument ('-p', '--pixelscale', type=float, default=-999.99, \
                     help='pixel scale in arcsec/pix')

# parsing arguments
args = parser.parse_args ()

# input parameters
skybg_flux = args.skybg
skybg_err  = args.skybg_err
star_flux  = args.star
star_mag   = args.mag
pixelscale = args.pixelscale

# check of input parameters
if (skybg_flux < 0.0):
    print ("Something is wrong with sky background level in ADU.")
    sys.exit ()
if (skybg_err < 0.0):
    print ("Something is wrong with error of sky background level in ADU.")
    sys.exit ()
if (star_flux < 0.0):
    print ("Something is wrong with net flux of star in ADU.")
    sys.exit ()
if (star_mag < 0.0):
    print ("Something is wrong with magnitude of star.")
    sys.exit ()
if (pixelscale < 0.0):
    print ("Something is wrong with pixel scale in arcsec/pix.")
    sys.exit ()
```

```
# calculation of sky background brightness in mag/arcsec^2
skybg_flux_per_sqarcsec = skybg_flux / pixelscale**2
skybg_mag = star_mag - 2.5 * numpy.log10 (skybg_flux_per_sqarcsec / star_flux)
skybg_err = 2.5 / numpy.log (10) * skybg_err / skybg_flux

# printing result
print ("sky background brightness = %6.3f +/- %6.3f mag/arcsec^2" \
       % (skybg_mag, skybg_err) )
```

Run the script, and show the result of sky background brightness calculation.

```
% chmod a+x ao2021_s16_09.py
% ./ao2021_s16_09.py -h
usage: ao2021_s16_09.py [-h] [-b SKYBG] [-e SKYBG_ERR] [-s STAR] [-m MAG]
                        [-p PIXELSCALE]

calculating sky background brightness

optional arguments:
  -h, --help            show this help message and exit
  -b SKYBG, --skybg SKYBG
                        sky background level in ADU
  -e SKYBG_ERR, --skybg-err SKYBG_ERR
                        error of sky background level in ADU
  -s STAR, --star STAR  net flux of standard star in ADU
  -m MAG, --mag MAG     apparent magnitude of standard star
  -p PIXELSCALE, --pixelscale PIXELSCALE
                        pixel scale in arcsec/pix

% ./ao2021_s16_09.py -b 174.911 -e 16.414 -s 281649.633641 -m 15.861 -p 0.3855
sky background brightness = 21.808 +/-  0.102 mag/arcsec^2
```

The sky background brightness in g'-band is estimated to be 21.8 ± 0.1 mag/arcsec^2.

9 For your training

Read followings.

1. "GitHub.com"
   - `https://docs.github.com/en/github`
2. "Dictionaries" in "The Python Tutorial"
   - `https://docs.python.org/3/tutorial/datastructures.html#dictionaries`
3. "Flexible Image Transport System"
   - `https://fits.gsfc.nasa.gov/`
4. "FITS File Handling" of Astropy package
   - `https://docs.astropy.org/en/stable/io/fits/index.html`
5. "Astrostatistics Tools" of Astropy package
   - `https://docs.astropy.org/en/stable/stats/index.html`
6. matplotlib.pyplot.hist

- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html

7. "Aperture Photometry" of Photutils package
   - https://photutils.readthedocs.io/en/stable/aperture.html
8. "Basic Photometry Techniques"
   - Da Costa
   - Astronomical CCD observing and reduction techniques, edited by Steve B. Howell. Published 1992 Astronomical Society of the Pacific, vol. 23, San Francisco, CA. ISBN 0-937707-42-4, LCCN 92-71172, p. 90.
   - https://ui.adsabs.harvard.edu/abs/1992ASPC...23...90D/abstract
9. "Estimating Lunar Phase Requirements"
   - Elias
   - 1994, NOAO Newsletter, Number 37.
   - https://www.noao.edu/noao/noaonews/mar94/art20.html
10. "A Model of the Brightness of Moonlight"
    - Krisciunas and Schaefer
    - 1991, PASP, 103, 1033.
    - https://ui.adsabs.harvard.edu/abs/1991PASP..103.1033K/abstract
11. "以M44疏散星團檢測鹿林前山之夜晚天光亮度"
    - 林宏欽 (Lin, Hung-Chin)
    - master thesis at National Central University
12. "Characteristics and Performance of the CCD Photometric System at Lulin Observatory"
    - Kinoshita et al.
    - 2005, ChJAA, 5, 315.
    - https://ui.adsabs.harvard.edu/abs/2005ChJAA...5..315K/abstract

10 Assignment

1. What are major contributions to night sky background brightness? Show references.
2. Use ADS (Astrophysics Data System, https://ui.adsabs.harvard.edu/) to search for papers reporting sky background brightness. Show sky background brightness in UBVRI and u'g'r'i'z' bands at major observatories in the world.
3. Use ADS (Astrophysics Data System, https://ui.adsabs.harvard.edu/) to search for papers reporting sky background brightness. Show sky background brightness at Lulin Observatory.
4. Estimate g'-band sky background brightness in mag/arcsec^2 using the FITS file lot_20210214_0186_df.fits. Show all the Python script you have written. Show the result of your estimate.
5. Estimate r'-band sky background brightness in mag/arcsec^2 using the FITS file lot_20210214_0187_df.fits or lot_20210214_0188_df.fits. Show all the Python script you have written. Show the result of your estimate.
6. Estimate i'-band sky background brightness in mag/arcsec^2 using the FITS file lot_20210214_0189_df.fits or lot_20210214_0190_df.fits. Show all the Python script you have written. Show the result of your estimate.
7. Use the formula in Krisciunas and Schaefer (1991) to estimate the night sky background brightness at zenith from your measurements in g', r', and i' images. Explain the method and results of your calculations.
8. Compare the results of your measurements with previously reported night sky background brightness at major astronomical sites in the world. Is the night sky background brightness dark compared to those at major astronomical sites?
9. Search for publicly available archived data of photometric standard star field. Download an image of photometric standard star field. Analyse the image, and estimate the night sky background brightness in mag/arcsec^2. Show the result of your measurement, name of the observing site, and conditions of the observations.