Advanced Astronomical Observations 2021
Session 04: Making Flatfield

Kinoshita Daisuke

17 March 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 make combined and normalised flatfield.

1 Downloading data

A set of FITS files for this session is placed at following location. Download the file. The size of the file is about 1420 MB.

- `https://s3b.astro.ncu.edu.tw/advobs_202102/data/data_ao2021_s05.tar.xz`

If you prefer to use a command-line tool, such as `curl`, then try following command.

```
% curl -k -o data_ao2021_s05.tar.xz \
? https://s3b.astro.ncu.edu.tw/advobs_202102/data/data_ao2021_s05.tar.xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1421M  100 1421M    0     0  1759k       0  0:13:47  0:13:47 --:--:-- 4163k
```

If you prefer to use a web browser, such as Firefox, then start a web browser and download the file.

2 Extracting data

The file you have downloaded is a compressed TAR archive file. To extract FITS files, try following command. 418 FITS files should be extracted from the archive file.

```
% tar xJvf data_ao2021_s05.tar.xz
x data_ao2021_s05/
x data_ao2021_s05/lot_20210215_0294.fits
x data_ao2021_s05/lot_20210215_0295.fits
x data_ao2021_s05/lot_20210215_0296.fits
x data_ao2021_s05/lot_20210215_0297.fits
x data_ao2021_s05/lot_20210215_0298.fits

.....

x data_ao2021_s05/lot_20210214_0787.fits
x data_ao2021_s05/lot_20210214_0788.fits
x data_ao2021_s05/lot_20210214_0789.fits
x data_ao2021_s05/lot_20210214_0790.fits
x data_ao2021_s05/lot_20210214_0791.fits
% ls data_ao2021_s05/*.fits | wc
     418     418   16302
```

If above command does not work on your computer, then try following.

```
% unxz -c data_ao2021_s04.tar.xz | tar xvf -
x data_ao2021_s05/
x data_ao2021_s05/lot_20210215_0294.fits
x data_ao2021_s05/lot_20210215_0295.fits
x data_ao2021_s05/lot_20210215_0296.fits
x data_ao2021_s05/lot_20210215_0297.fits
x data_ao2021_s05/lot_20210215_0298.fits

.....

x data_ao2021_s05/lot_20210214_0787.fits
x data_ao2021_s05/lot_20210214_0788.fits
x data_ao2021_s05/lot_20210214_0789.fits
x data_ao2021_s05/lot_20210214_0790.fits
x data_ao2021_s05/lot_20210214_0791.fits
% ls data_ao2021_s05/*.fits | wc
     418     418   16302
```

If above command fails, you probably do not have XZ Utils. If you do not have XZ Utils, visit following website (Fig. 1) and install XZ Utils.

- `https://tukaani.org/xz/`

If you are not familiar to pipes of Unix shells, try following.

```
% ls -l data_ao2021_s05.tar.xz
-rw-r--r--  1 daisuke  taiwan  1490181256 Mar 16 14:07 data_ao2021_s05.tar.xz
% unxz data_ao2021_s05.tar.xz
% ls -l data_ao2021_s05.tar
-rw-r--r--  1 daisuke  taiwan  3509435904 Mar 16 14:07 data_ao2021_s05.tar
% tar xvf data_ao2021_s05.tar
x data_ao2021_s05/
x data_ao2021_s05/lot_20210215_0294.fits
x data_ao2021_s05/lot_20210215_0295.fits
x data_ao2021_s05/lot_20210215_0296.fits
x data_ao2021_s05/lot_20210215_0297.fits
x data_ao2021_s05/lot_20210215_0298.fits
```

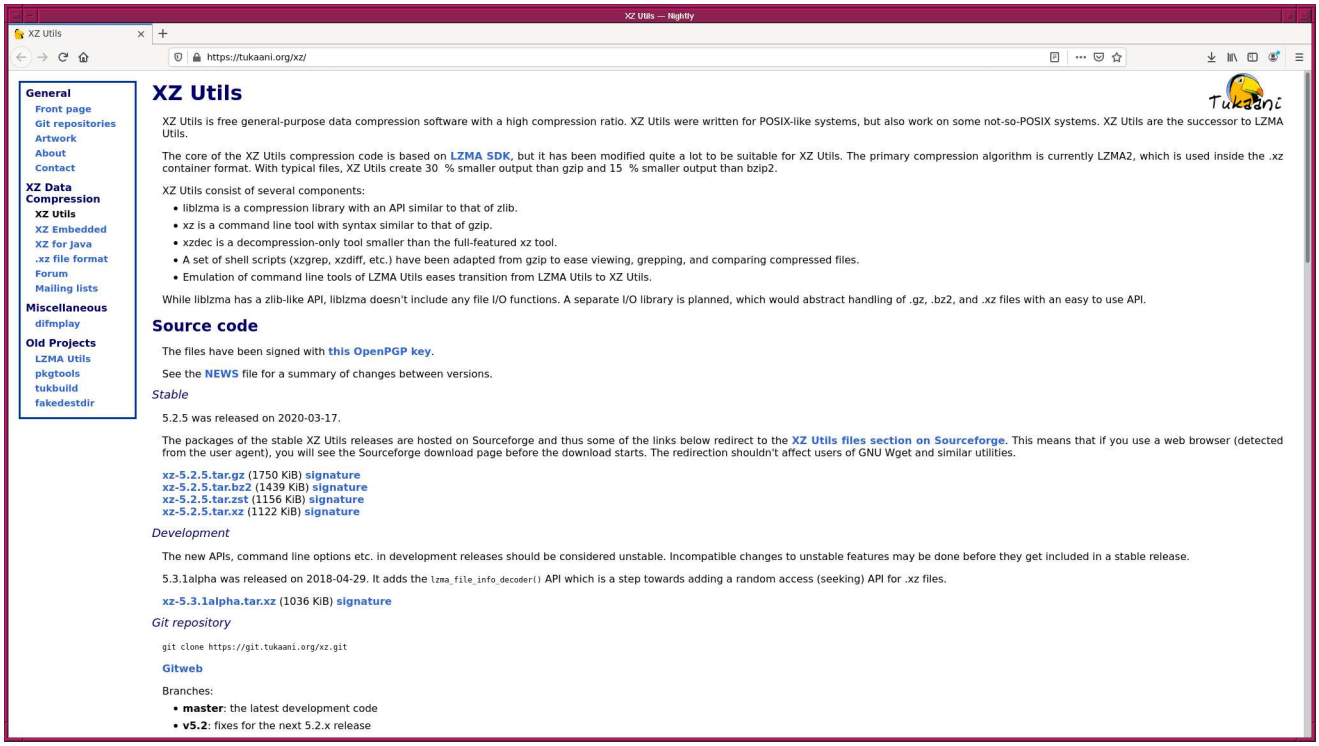

Figure 1: The website of XZ Utils.

```
.....

x data_ao2021_s05/lot_20210214_0787.fits
x data_ao2021_s05/lot_20210214_0788.fits
x data_ao2021_s05/lot_20210214_0789.fits
x data_ao2021_s05/lot_20210214_0790.fits
x data_ao2021_s05/lot_20210214_0791.fits
% ls data_ao2021_s05/*.fits | wc
     418     418   16302
```

3 Checking data

Read FITS files, and check the header part of FITS files.

Python Code 1: ao2021_s05_01.py

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

# importing argparse module
import argparse

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Generating a simple observing log'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
```

```
default_keyword = 'DATE-OBS,TIME-OBS,IMAGETYP,EXPTIME,FILTER'
parser.add_argument ('-k', '--keyword', default=default_keyword, \
                     help='a list of keyword to check')
parser.add_argument ('files', nargs='+', help='FITS files')

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

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

# a list of keywords
list_keyword = keyword.split (',')

# processing files
for file in files:
    # if the extension of the file is not '.fits', then skip
    if (file[-5:] != '.fits'):
        continue

    # file name
    path = file.split ('/')
    filename = path[-1]

    # opening FITS file
    hdu_list = astropy.io.fits.open (file)

    # primary HDU
    hdu0 = hdu_list[0]

    # header of primary HDU
    header0 = hdu0.header

    # closing FITS file
    hdu_list.close ()

    # gathering information from FITS header
    record = filename
    for key in list_keyword:
        # obtaining a value for given keyword
        if key in header0:
            value = str (header0[key])
        else:
            value = "__NONE__"
        # appending the value to the string "record"
        if (key == 'DATE-OBS'):
            record += "  %-10s" % value
        elif (key == 'TIME-OBS'):
            record += "  %-8s" % value
        elif (key == 'IMAGETYP'):
            record += "  %-5s" % value
        elif (key == 'EXPTIME'):
            record += "  %6.1f" % float (value)
        elif (key == 'FILTER'):
            record += "  %-16s" % value
        else:
            record += "  %s" % value
```

```
    # printing information
    print (record)
```

Run the script, and show the observing log for data of this session.

```
% chmod a+x ao2021_s05_01.py
% ./ao2021_s05_01.py data_ao2021_s05/*.fits
lot_20210214_0394.fits  2021-02-14  21:35:45  FLAT     45.0  gp_Astrodon_2019
lot_20210214_0395.fits  2021-02-14  21:36:39  FLAT     45.0  V_319142
lot_20210214_0396.fits  2021-02-14  21:37:34  FLAT     45.0  rp_Astrodon_2019
lot_20210214_0397.fits  2021-02-14  21:38:27  FLAT     45.0  R_10349
lot_20210214_0398.fits  2021-02-14  21:39:22  FLAT     45.0  ip_Astrodon_2019
lot_20210214_0399.fits  2021-02-14  21:40:20  FLAT     45.0  gp_Astrodon_2019
lot_20210214_0400.fits  2021-02-14  21:41:14  FLAT     45.0  V_319142
lot_20210214_0401.fits  2021-02-14  21:42:08  FLAT     45.0  rp_Astrodon_2019
lot_20210214_0402.fits  2021-02-14  21:43:02  FLAT     45.0  R_10349
lot_20210214_0403.fits  2021-02-14  21:43:57  FLAT     45.0  ip_Astrodon_2019

.....

lot_20210215_0464.fits  2021-02-15  21:15:27  DARK      5.0  __NONE__
lot_20210215_0465.fits  2021-02-15  21:15:39  FLAT      5.0  ip_Astrodon_2019
lot_20210215_0466.fits  2021-02-15  21:15:51  BIAS      0.0  __NONE__
lot_20210215_0467.fits  2021-02-15  21:15:56  DARK      5.0  __NONE__
lot_20210215_0468.fits  2021-02-15  21:16:08  FLAT      5.0  ip_Astrodon_2019
lot_20210215_0469.fits  2021-02-15  21:16:20  BIAS      0.0  __NONE__
lot_20210215_0470.fits  2021-02-15  21:16:25  DARK      5.0  __NONE__
lot_20210215_0471.fits  2021-02-15  21:16:37  FLAT      5.0  ip_Astrodon_2019
lot_20210215_0472.fits  2021-02-15  21:16:49  BIAS      0.0  __NONE__
lot_20210215_0473.fits  2021-02-15  21:16:54  DARK      5.0  __NONE__
```

The data for this session are flatfield, dark, and bias frames. Flatfield frames taken on 14/Feb/2021 are twilight flatfield, and flatfield frames taken on 15/Feb/2021 are dome flatfield.

4 Making dome flatfield

We first construct g'-band dome flatfield from the data taken on 15/Feb/2021.

4.1 Searching data for dome flatfield

Show g'-band flatfield frames taken on 15/Feb/2021 using previous script.

```
% ./ao2021_s05_01.py data_ao2021_s05/*.fits | grep 02-15 | grep gp_Astrodon_2019
lot_20210215_0294.fits  2021-02-15  20:29:19  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0297.fits  2021-02-15  20:30:43  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0300.fits  2021-02-15  20:32:02  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0303.fits  2021-02-15  20:33:19  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0306.fits  2021-02-15  20:34:39  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0309.fits  2021-02-15  20:35:58  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0312.fits  2021-02-15  20:37:16  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0315.fits  2021-02-15  20:38:33  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0318.fits  2021-02-15  20:39:53  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0321.fits  2021-02-15  20:41:12  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0324.fits  2021-02-15  20:42:31  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0327.fits  2021-02-15  20:43:48  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0330.fits  2021-02-15  20:45:08  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0333.fits  2021-02-15  20:46:26  FLAT     30.0  gp_Astrodon_2019
```

```
lot_20210215_0336.fits  2021-02-15  20:47:44  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0339.fits  2021-02-15  20:49:04  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0342.fits  2021-02-15  20:50:24  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0345.fits  2021-02-15  20:51:43  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0348.fits  2021-02-15  20:53:01  FLAT     30.0  gp_Astrodon_2019
lot_20210215_0351.fits  2021-02-15  20:54:21  FLAT     30.0  gp_Astrodon_2019
```

The exposure time of g'-band flatfield frames taken on 15/Feb/2021 is 30 sec.
If you have not used `grep` command before, try following command to learn about the command (Fig. 2).

```
% man grep
```

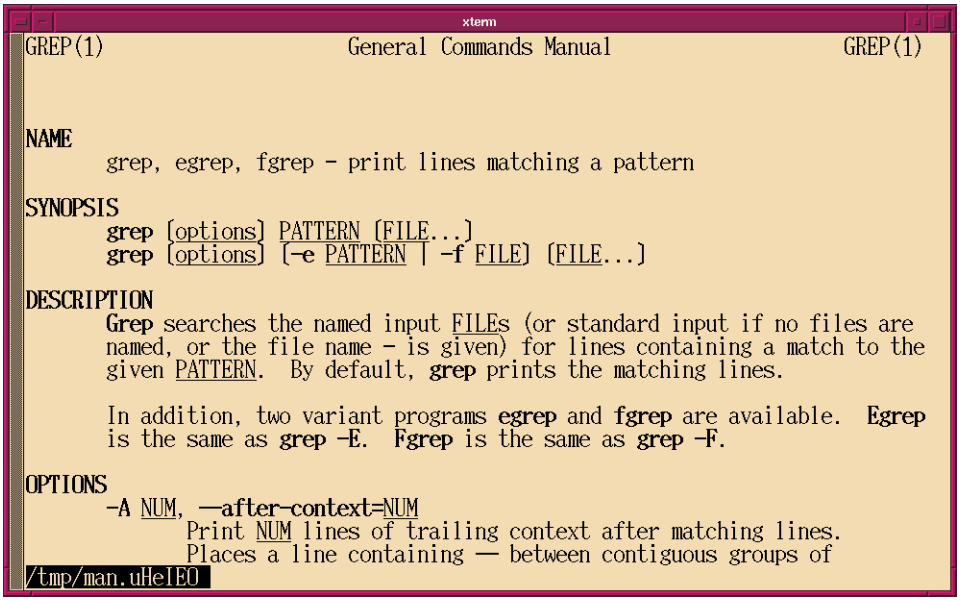

Figure 2: The manual page of the command `grep`.

Next, search for dark frames of 30-sec exposure time.

```
% ./ao2021_s05_01.py data_ao2021_s05/*.fits | grep 02-15 | grep DARK | grep 30.0
lot_20210215_0296.fits  2021-02-15  20:30:06  DARK     30.0  __NONE__
lot_20210215_0299.fits  2021-02-15  20:31:24  DARK     30.0  __NONE__
lot_20210215_0302.fits  2021-02-15  20:32:43  DARK     30.0  __NONE__
lot_20210215_0305.fits  2021-02-15  20:34:02  DARK     30.0  __NONE__
lot_20210215_0308.fits  2021-02-15  20:35:20  DARK     30.0  __NONE__
lot_20210215_0311.fits  2021-02-15  20:36:39  DARK     30.0  __NONE__
lot_20210215_0314.fits  2021-02-15  20:37:56  DARK     30.0  __NONE__
lot_20210215_0317.fits  2021-02-15  20:39:16  DARK     30.0  __NONE__
lot_20210215_0320.fits  2021-02-15  20:40:34  DARK     30.0  __NONE__
lot_20210215_0323.fits  2021-02-15  20:41:55  DARK     30.0  __NONE__
lot_20210215_0326.fits  2021-02-15  20:43:12  DARK     30.0  __NONE__
lot_20210215_0329.fits  2021-02-15  20:44:30  DARK     30.0  __NONE__
lot_20210215_0332.fits  2021-02-15  20:45:49  DARK     30.0  __NONE__
lot_20210215_0335.fits  2021-02-15  20:47:08  DARK     30.0  __NONE__
lot_20210215_0338.fits  2021-02-15  20:48:26  DARK     30.0  __NONE__
lot_20210215_0341.fits  2021-02-15  20:49:47  DARK     30.0  __NONE__
lot_20210215_0344.fits  2021-02-15  20:51:06  DARK     30.0  __NONE__
lot_20210215_0347.fits  2021-02-15  20:52:25  DARK     30.0  __NONE__
lot_20210215_0350.fits  2021-02-15  20:53:43  DARK     30.0  __NONE__
lot_20210215_0353.fits  2021-02-15  20:55:03  DARK     30.0  __NONE__
```

4.2 Examining data

Read FITS files, and show statistical information of pixel values.

Python Code 2: ao2021_s05_02.py

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

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing scipy module
import scipy.stats

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Calculating statistical values'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_rejection = ['none', 'sigclip']
list_datatype = ['LIGHT', 'FLAT', 'DARK', 'BIAS']
list_filter = ['gp_Astrodon_2019', 'rp_Astrodon_2019', 'ip_Astrodon_2019', \
               'V_319142', 'R_10349', '__NONE__']
parser.add_argument ('-d', '--datatype', choices=list_datatype, \
                     default='LIGHT', help='accepted data type')
parser.add_argument ('-e', '--exptime', type=float, \
                     default=5.0, help='accepted exposure time')
parser.add_argument ('-f', '--filter', choices=list_filter, \
                     default='__NONE__', help='accepted data type')
parser.add_argument ('-r', '--rejection', choices=list_rejection, \
                     default='none', help='outlier rejection algorithm')
parser.add_argument ('-t', '--threshold', type=float, default=4.0, \
                     help='rejection threshold in sigma')
parser.add_argument ('files', nargs='+', help='FITS files')

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

# input parameters
files     = args.files
rejection = args.rejection
threshold = args.threshold
datatype  = args.datatype
exptime   = args.exptime
filter    = args.filter

# printing header
print ("%s" % '-' * 79)
print ("%-25s %8s %8s %8s %8s %8s %8s" \
       % ("file name", "n_pix", "mean", "median", "stddev", "min", "max") )
print ("%s" % '=' * 79)

# processing files
for file in files:
```

```
    # if the extension of the file is not '.fits', then skip
    if (file[-5:] != '.fits'):
        continue

    # file name
    path = file.split ('/')
    filename = path[-1]

    # opening FITS file
    hdu_list = astropy.io.fits.open (file)

    # primary HDU
    hdu0 = hdu_list[0]

    # header of primary HDU
    header0 = hdu0.header

    # if the FITS file is not what you want, then skip
    if ('FILTER' in header0):
        if not ( (header0['IMAGETYP'] == datatype) \
                 and (header0['EXPTIME'] == exptime) \
                 and (header0['FILTER'] == filter) ):
            continue
    else:
        if not ( (header0['IMAGETYP'] == datatype) \
                 and (header0['EXPTIME'] == exptime) ):
            continue

    # flattened image data of primary HDU
    data0 = hdu0.data.flatten ()

    # closing FITS file
    hdu_list.close ()

    # if rejection algorithm is used, then do rejection check
    if (rejection == 'sigclip'):
        # sigma clipping using scipy module
        clipped, lower, upper \
            = scipy.stats.sigmaclip (data0, low=threshold, high=threshold)
    elif (rejection == 'none'):
        clipped = data0

    # calculation of statistical values
    n_pix  = len (clipped)
    mean   = numpy.nanmean (clipped)
    median = numpy.nanmedian (clipped)
    stddev = numpy.nanstd (clipped)
    vmin   = numpy.nanmin (clipped)
    vmax   = numpy.nanmax (clipped)

    # printing results
    print ("%-25s %8d %8.2f %8.2f %8.2f %8.2f %8.2f" \
           % (filename, n_pix, mean, median, stddev, vmin, vmax) )

# printing footer
print ("%s" % '-' * 79)
```

Run the script and show the mean values of g'-band flatfield frames of 30-sec exposure time.

```
% chmod a+x ao2021\_s05\_02.py
% ./ao2021_s05_02.py -d FLAT -e 30 -f gp_Astrodon_2019 -r sigclip -t 5 */*.fits
-------------------------------------------------------------------------------
file name                      n_pix     mean   median   stddev      min      max
===============================================================================
lot_20210215_0294.fits     4194208 13491.88 13503.00   301.68 11991.00 14681.00
lot_20210215_0297.fits     4194205 13830.35 13842.00   308.52 12289.00 15015.00
lot_20210215_0300.fits     4194195 13494.44 13505.00   300.62 11993.00 14986.00
lot_20210215_0303.fits     4194215 13449.24 13460.00   299.54 11952.00 14600.00
lot_20210215_0306.fits     4194205 13993.13 14005.00   312.42 12434.00 15481.00
lot_20210215_0309.fits     4194209 13293.99 13304.00   295.63 11822.00 14452.00
lot_20210215_0312.fits     4194197 13629.72 13640.00   303.12 12118.00 14852.00
lot_20210215_0315.fits     4194216 12784.88 12795.00   284.92 11366.00 14074.00
lot_20210215_0318.fits     4194211 12545.28 12555.00   279.70 11147.00 13738.00
lot_20210215_0321.fits     4194212 13489.72 13500.00   300.20 11989.00 14946.00
lot_20210215_0324.fits     4194202 13062.46 13073.00   291.49 11607.00 14487.00
lot_20210215_0327.fits     4194202 13984.48 13996.00   311.70 12426.00 15493.00
lot_20210215_0330.fits     4194197 15554.28 15567.00   344.35 13834.00 16926.00
lot_20210215_0333.fits     4194205 15884.34 15898.00   352.67 14122.00 17615.00
lot_20210215_0336.fits     4194197 14711.01 14723.00   327.26 13078.00 15933.00
lot_20210215_0339.fits     4194192 15929.30 15942.00   352.41 14172.00 17485.00
lot_20210215_0342.fits     4194198 15462.53 15476.00   343.15 13748.00 17168.00
lot_20210215_0345.fits     4194202 15715.51 15730.00   351.32 13968.00 17471.00
lot_20210215_0348.fits     4194210 14118.02 14131.00   316.58 12539.00 15426.00
lot_20210215_0351.fits     4194186 14618.93 14632.00   326.45 12991.00 16245.00
-------------------------------------------------------------------------------
```

The mean pixel values of raw data of 30-sec g'-band dome flatfield frames are $\sim$ 14,000 ADU.
Next, examine mean values of 30-sec dark frames.

```
% ./ao2021_s05_02.py -d DARK -e 30 -r sigclip -t 5 */*.fits
-------------------------------------------------------------------------------
file name                      n_pix     mean   median   stddev      min      max
===============================================================================
lot_20210215_0296.fits     4194268   605.91   606.00     8.07   566.00   646.00
lot_20210215_0299.fits     4194261   606.10   606.00     8.07   566.00   646.00
lot_20210215_0302.fits     4194238   606.07   606.00     8.07   566.00   646.00
lot_20210215_0305.fits     4194203   605.56   606.00     8.07   566.00   645.00
lot_20210215_0308.fits     4194248   605.70   606.00     8.08   566.00   645.00
lot_20210215_0311.fits     4194250   605.86   606.00     8.08   566.00   646.00
lot_20210215_0314.fits     4194253   606.17   606.00     8.08   567.00   646.00
lot_20210215_0317.fits     4194283   605.83   606.00     8.09   566.00   646.00
lot_20210215_0320.fits     4194210   605.58   606.00     8.09   566.00   646.00
lot_20210215_0323.fits     4194241   606.83   607.00     8.06   567.00   647.00
lot_20210215_0326.fits     4194244   605.55   606.00     8.07   566.00   645.00
lot_20210215_0329.fits     4194240   606.99   607.00     8.08   567.00   647.00
lot_20210215_0332.fits     4194269   605.79   606.00     8.10   566.00   646.00
lot_20210215_0335.fits     4194280   606.35   606.00     8.08   566.00   646.00
lot_20210215_0338.fits     4194232   606.11   606.00     8.10   566.00   646.00
lot_20210215_0341.fits     4194214   606.60   607.00     8.09   567.00   647.00
lot_20210215_0344.fits     4194212   605.76   606.00     8.09   566.00   646.00
lot_20210215_0347.fits     4194156   606.07   606.00     8.09   567.00   646.00
lot_20210215_0350.fits     4194214   606.53   606.00     8.08   567.00   646.00
lot_20210215_0353.fits     4194251   605.53   606.00     8.08   566.00   645.00
-------------------------------------------------------------------------------
```

4.3 Checking a raw flatfield frame

Choose a raw flatfield frame, and visualise it.

Python Code 3: ao2021_s05_03.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

# construction of parser object
desc = 'Reading a FITS file and making a PNG file'
parser = argparse.ArgumentParser (description=desc)

# colour maps
list_cmap = ['viridis', 'plasma', 'inferno', 'magma', 'cividis', \
             'gray', 'bone', 'pink', 'cool', 'hot', \
             'spring', 'summer', 'autumn', 'winter']

# adding arguments
parser.add_argument ('-a', '--min', type=float, default=0.0, \
                     help='minimum pixel value')
parser.add_argument ('-b', '--max', type=float, default=65535.0, \
                     help='maximum pixel value')
parser.add_argument ('-c', '--cmap', default='hot', choices=list_cmap, \
                     help='maximum pixel value')
parser.add_argument ('-i', '--input', default='test.fits', \
                     help='intput FITS file')
parser.add_argument ('-o', '--output', default='test.png', \
                     help='output image file')

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

# input FITS file
file_input  = args.input
file_output = args.output
vmin        = args.min
vmax        = args.max
cmap        = args.cmap

# if input file is not a FITS file, then skip
if not (file_input[-5:] == '.fits'):
    # printing a message
    print ("Error: input file must be a FITS file!")
    # exit
    sys.exit ()

# if output file is not either PNG, PDF, or PS, then skip
```

```
if not ( (file_output[-4:] == '.png') or (file_output[-4:] == '.pdf') \
         or (file_output[-3:] == '.ps') ):
    # printing a message
    print ("Error: output file must be a PNG or PDF or PS!")
    # exit
    sys.exit ()

# opening FITS file
hdu_list = astropy.io.fits.open (file_input)

# primary HDU
hdu0 = hdu_list[0]

# reading header
header0 = hdu0.header

# reading data
data0 = hdu0.data

# closing FITS file
hdu_list.close ()

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

# axes
ax.set_title (file_input)
ax.set_xlabel ('X [pixel]')
ax.set_ylabel ('Y [pixel]')

# plotting image
im = ax.imshow (data0, origin='lower', cmap=cmap, vmin=vmin, vmax=vmax)
fig.colorbar (im)

# saving file
print ("%s ==> %s" % (file_input, file_output) )
fig.savefig (file_output, dpi=450)
```

Run the script and generate a PNG file.

```
% chmod a+x ao2021_s05_03.py
% ./ao2021_s05_03.py -a 11000 -b 16000 \
? -i data_ao2021_s05/lot_20210215_0294.fits -o 20210215_0294_raw.png
data_ao2021_s05/lot_20210215_0294.fits ==> 20210215_0294_raw.png
% ls -l 20210215_0294_raw.png
-rw-r--r--  1 daisuke  taiwan  3111917 Mar 16 15:48 20210215_0294_raw.png
```

Display the PNG image. (Fig. 3)

```
% feh -dF 20210215_0294_raw.png
```

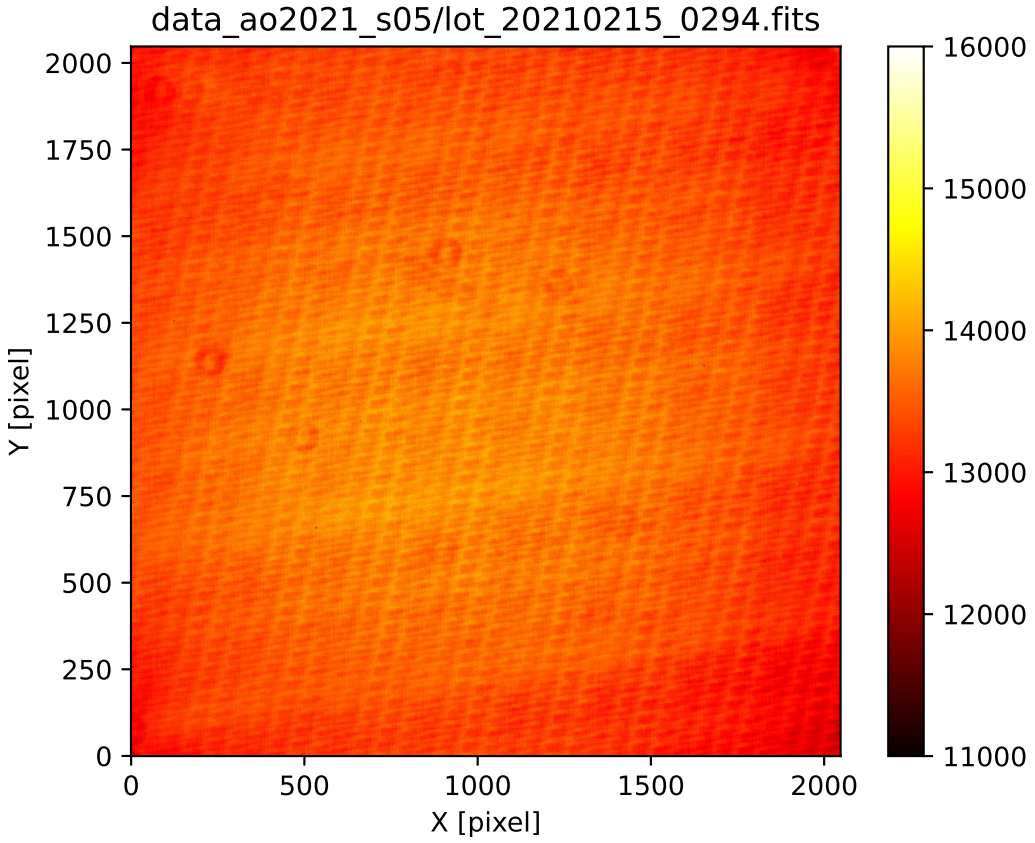

Figure 3: The raw data of flatfield frame `lot_20210215_0294.fits`.

4.4 Combining dark frames

Now, we combine dark frames of 30-sec exposure time using sigma clipping algorithm for later dark subtraction.

Python Code 4: ao2021_s05_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.io.fits
import astropy.stats

# importing datetime module
import datetime

# construction of parser object
desc = 'Combining images'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_rejection = ['none', 'sigclip']
list_cenfunc   = ['mean', 'median']
list_datatype  = ['LIGHT', 'FLAT', 'DARK', 'BIAS']
list_filter    = ['gp_Astrodon_2019', 'rp_Astrodon_2019', 'ip_Astrodon_2019', \
                  'V_319142', 'R_10349', '__NONE__']
parser.add_argument ('-d', '--datatype', choices=list_datatype, \
                     default='LIGHT', help='accepted data type')
parser.add_argument ('-e', '--exptime', type=float, \
                     default=5.0, help='accepted exposure time')
parser.add_argument ('-f', '--filter', choices=list_filter, \
                     default='__NONE__', help='accepted data type')
parser.add_argument ('-r', '--rejection', choices=list_rejection, \
                     default='none', help='outlier rejection algorithm')
parser.add_argument ('-t', '--threshold', type=float, default=5.0, \
                     help='rejection threshold in sigma')
parser.add_argument ('-n', '--maxiters', type=int, default=10, \
                     help='maximum number of iterations')
parser.add_argument ('-c', '--cenfunc', choices=list_cenfunc, \
                     default='median', help='method to estimate centre value')
parser.add_argument ('-o', '--output', default='combined.fits', \
                     help='output FITS file')
parser.add_argument ('files', nargs='+', help='input FITS files')

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

# parameters given by command-line arguments
file_input  = args.files
file_output = args.output
rejection   = args.rejection
threshold   = args.threshold
```

```
cenfunc     = args.cenfunc
maxiters    = args.maxiters
datatype  = args.datatype
exptime   = args.exptime
filter    = args.filter

# command name
command = sys.argv[0]

# checking number of intput FITS files
if ( len (file_input) < 2 ):
    # if the number of input files is less than 2, then stop the script
    print ("Number of input files must be 2 or larger!")
    # exit the script
    sys.exit ()

# checking input files
for file_fits in file_input:
    # if the file is not a FITS file, then stop the script
    if not (file_fits[-5:] == '.fits'):
        # printing error message
        print ("Input files must be FITS files!")
        print ("The file \"%s\" is not a FITS file!" % file_fits)
        # exit the script
        sys.exit ()

# checking output file
# if the file is not a FITS file, then stop the script
if not (file_output[-5:] == '.fits'):
    # printing error message
    print ("Output file must be FITS files!")
    # exit the script
    sys.exit ()

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

# parameter for counting
i = 0

# printing information
print ("Data criteria:")
print ("  data type     = %s" % datatype)
print ("  exposure time = %f sec" % exptime)
print ("  filter        = %s" % filter)
print ("List of files to be combined:")

# a list for file names to be combined
file_selected = []

# reading FITS files and constructing a data cube
for file_fits in file_input:
    # opening FITS file
    hdu_list = astropy.io.fits.open (file_fits)

    # primary HDU
    hdu0 = hdu_list[0]

    # reading header
```

```
    header0 = hdu0.header

    # if the FITS file is not what you want, then skip
    if ('FILTER' in header0):
        if not ( (header0['IMAGETYP'] == datatype) \
                 and (header0['EXPTIME'] == exptime) \
                 and (header0['FILTER'] == filter) ):
            # closing FITS file
            hdu_list.close ()
            continue
    else:
        if not ( (header0['IMAGETYP'] == datatype) \
                 and (header0['EXPTIME'] == exptime) ):
            # closing FITS file
            hdu_list.close ()
            continue

    file_selected.append (file_fits)

    # copying header only for the first FITS file
    if (i == 0):
        header = header0

    # reading data
    data0 = hdu0.data

    # closing FITS file
    hdu_list.close ()

    # constructing a data cube
    if (i == 0):
        tmp0 = data0
    elif (i == 1):
        cube = numpy.concatenate ( ([tmp0], [data0]), axis=0 )
    else:
        cube = numpy.concatenate ( (cube, [data0]), axis=0 )

    # incrementing the parameter "i"
    i += 1

    # printing information
    print ("  %s" % file_fits)

# printing information
print ("Output file name: %s" % file_output)
print ("Parameters:")
print ("  rejection = %s" % rejection)
print ("  threshold = %f" % threshold)
print ("  maxiters  = %d" % maxiters)
print ("  cenfunc   = %s" % cenfunc)

# combining images into a single co-added image
if (rejection == 'sigclip'):
    # combining using sigma clipping
    combined, median, stddev \
        = astropy.stats.sigma_clipped_stats (cube, sigma=threshold, \
                                             maxiters=maxiters, \
                                             cenfunc=cenfunc, stdfunc='std', \
                                             axis=0)
```

```
elif (rejection == 'none'):
    # combining using simple mean
    combined = numpy.nanmean (cube, axis=0)

# adding comments to the header
header['history'] = "FITS file created by the command \"%s\"" % (command)
header['history'] = "Updated on %s" % (now)
header['comment'] = "List of combined files:"
for fits in file_selected:
    header['comment'] = "  %s" % (fits)
header['comment'] = "Options given:"
header['comment'] = "  rejection = %s" % (rejection)
header['comment'] = "  threshold = %f sigma" % (threshold)
header['comment'] = "  maxiters  = %d" % (maxiters)
header['comment'] = "  cenfunc   = %s" % (cenfunc)

# writing a new FITS file
astropy.io.fits.writeto (file_output, combined, header=header)
```

Execute the script, and combine dark frames.

```
% chmod a+x ao2021_s05_04.py
% ./ao2021_s05_04.py -d DARK -e 30 -r sigclip -t 4 -o dark_0030.fits \
? data_ao2021_s05/*.fits
Data criteria:
  data type     = DARK
  exposure time = 30.000000 sec
  filter        = __NONE__
List of files to be combined:
  data_ao2021_s05/lot_20210215_0296.fits
  data_ao2021_s05/lot_20210215_0299.fits
  data_ao2021_s05/lot_20210215_0302.fits
  data_ao2021_s05/lot_20210215_0305.fits
  data_ao2021_s05/lot_20210215_0308.fits
  data_ao2021_s05/lot_20210215_0311.fits
  data_ao2021_s05/lot_20210215_0314.fits
  data_ao2021_s05/lot_20210215_0317.fits
  data_ao2021_s05/lot_20210215_0320.fits
  data_ao2021_s05/lot_20210215_0323.fits
  data_ao2021_s05/lot_20210215_0326.fits
  data_ao2021_s05/lot_20210215_0329.fits
  data_ao2021_s05/lot_20210215_0332.fits
  data_ao2021_s05/lot_20210215_0335.fits
  data_ao2021_s05/lot_20210215_0338.fits
  data_ao2021_s05/lot_20210215_0341.fits
  data_ao2021_s05/lot_20210215_0344.fits
  data_ao2021_s05/lot_20210215_0347.fits
  data_ao2021_s05/lot_20210215_0350.fits
  data_ao2021_s05/lot_20210215_0353.fits
Output file name: dark_0030.fits
Parameters:
  rejection = sigclip
  threshold = 4.000000
  maxiters  = 10
  cenfunc   = median
% ls -l dark_0030.fits
-rw-r--r--  1 daisuke  taiwan  33595200 Mar 16 16:11 dark_0030.fits
```

Make a Python script to show the header part of the FITS file.

Python Code 5: ao2021_s05_05.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Printing FITS header'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('file', nargs=1, default='test.fits', \
                     help='input fits file')

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

# parameters given by command-line arguments
file_input  = args.file

# checking input file
# if the file is not a FITS file, then stop the script
for fits in file_input:
    if not (fits[-5:] == '.fits'):
        # printing error message
        print ("Input file must be FITS files!")
        # exit the script
        sys.exit ()

    # opening FITS file
    hdu_list = astropy.io.fits.open (fits)

    # primary HDU
    hdu0 = hdu_list[0]

    # reading header
    header0 = hdu0.header

    # closing FITS file
    hdu_list.close ()

    # printing header
    print (repr (header0))
```

Run the script and show the header of newly created FITS file.

```
% chmod a+x ao2021_s05_05.py
% ./ao2021_s05_05.py dark_0030.fits
SIMPLE  =                    T / conforms to FITS standard
```

```
BITPIX  =                  -64 / array data type
NAXIS   =                    2 / number of array dimensions
NAXIS1  =                 2048
NAXIS2  =                 2048

.....

HISTORY FITS file created by the command "./ao2021_s05_04.py"
HISTORY Updated on 2021-03-16T16:15:26.697600
COMMENT List of combined files:
COMMENT   data_ao2021_s05/lot_20210215_0296.fits
COMMENT   data_ao2021_s05/lot_20210215_0299.fits
COMMENT   data_ao2021_s05/lot_20210215_0302.fits
COMMENT   data_ao2021_s05/lot_20210215_0305.fits
COMMENT   data_ao2021_s05/lot_20210215_0308.fits
COMMENT   data_ao2021_s05/lot_20210215_0311.fits
COMMENT   data_ao2021_s05/lot_20210215_0314.fits
COMMENT   data_ao2021_s05/lot_20210215_0317.fits
COMMENT   data_ao2021_s05/lot_20210215_0320.fits
COMMENT   data_ao2021_s05/lot_20210215_0323.fits
COMMENT   data_ao2021_s05/lot_20210215_0326.fits
COMMENT   data_ao2021_s05/lot_20210215_0329.fits
COMMENT   data_ao2021_s05/lot_20210215_0332.fits
COMMENT   data_ao2021_s05/lot_20210215_0335.fits
COMMENT   data_ao2021_s05/lot_20210215_0338.fits
COMMENT   data_ao2021_s05/lot_20210215_0341.fits
COMMENT   data_ao2021_s05/lot_20210215_0344.fits
COMMENT   data_ao2021_s05/lot_20210215_0347.fits
COMMENT   data_ao2021_s05/lot_20210215_0350.fits
COMMENT   data_ao2021_s05/lot_20210215_0353.fits
COMMENT Options given:
COMMENT   rejection = sigclip
COMMENT   threshold = 4.000000 sigma
COMMENT   maxiters  = 10
COMMENT   cenfunc   = median
```

Show the statistical information of pixel values.

```
% ./ao2021_s05_02.py -d DARK -e 30 dark_0030.fits */*.fits
--------------------------------------------------------------------------------
file name                      n_pix     mean   median   stddev      min      max
================================================================================
dark_0030.fits               4194304   606.05   606.05     1.93   590.85   944.05
lot_20210215_0296.fits       4194304   605.92   606.00     9.28   564.00  9585.00
lot_20210215_0299.fits       4194304   606.11   606.00     9.74   566.00 10869.00
lot_20210215_0302.fits       4194304   606.08   606.00     8.33   566.00  3395.00
lot_20210215_0305.fits       4194304   605.57   606.00     8.46   565.00  2603.00
lot_20210215_0308.fits       4194304   605.71   606.00    15.09   561.00 26289.00
lot_20210215_0311.fits       4194304   605.87   606.00     8.87   566.00  7238.00
lot_20210215_0314.fits       4194304   606.17   606.00     8.21   567.00  2375.00
lot_20210215_0317.fits       4194304   605.83   606.00     8.12   566.00  1613.00
lot_20210215_0320.fits       4194304   605.59   606.00    10.82   564.00 13088.00
lot_20210215_0323.fits       4194304   606.84   607.00     8.49   567.00  2915.00
lot_20210215_0326.fits       4194304   605.56   606.00     8.43   565.00  3913.00
lot_20210215_0329.fits       4194304   606.99   607.00     8.44   562.00  3585.00
lot_20210215_0332.fits       4194304   605.79   606.00     8.44   566.00  3729.00
lot_20210215_0335.fits       4194304   606.35   606.00     8.12   565.00  1721.00
lot_20210215_0338.fits       4194304   606.12   606.00     8.40   564.00  2236.00
```

```
lot_20210215_0341.fits      4194304   606.61   607.00     9.15   561.00  7860.00
lot_20210215_0344.fits      4194304   605.77   606.00     8.48   564.00  2949.00
lot_20210215_0347.fits      4194304   606.09   606.00    10.65   565.00  8563.00
lot_20210215_0350.fits      4194304   606.54   606.00     8.53   566.00  3295.00
lot_20210215_0353.fits      4194304   605.54   606.00     8.37   563.00  3419.00
-------------------------------------------------------------------------------
```

Generate and show the PNG image. (Fig. 4)

```
% ./ao2021_s05_03.py -a 595 -b 615 -i dark_0030.fits -o dark_0030.png
% feh -dF dark_0030.png
```

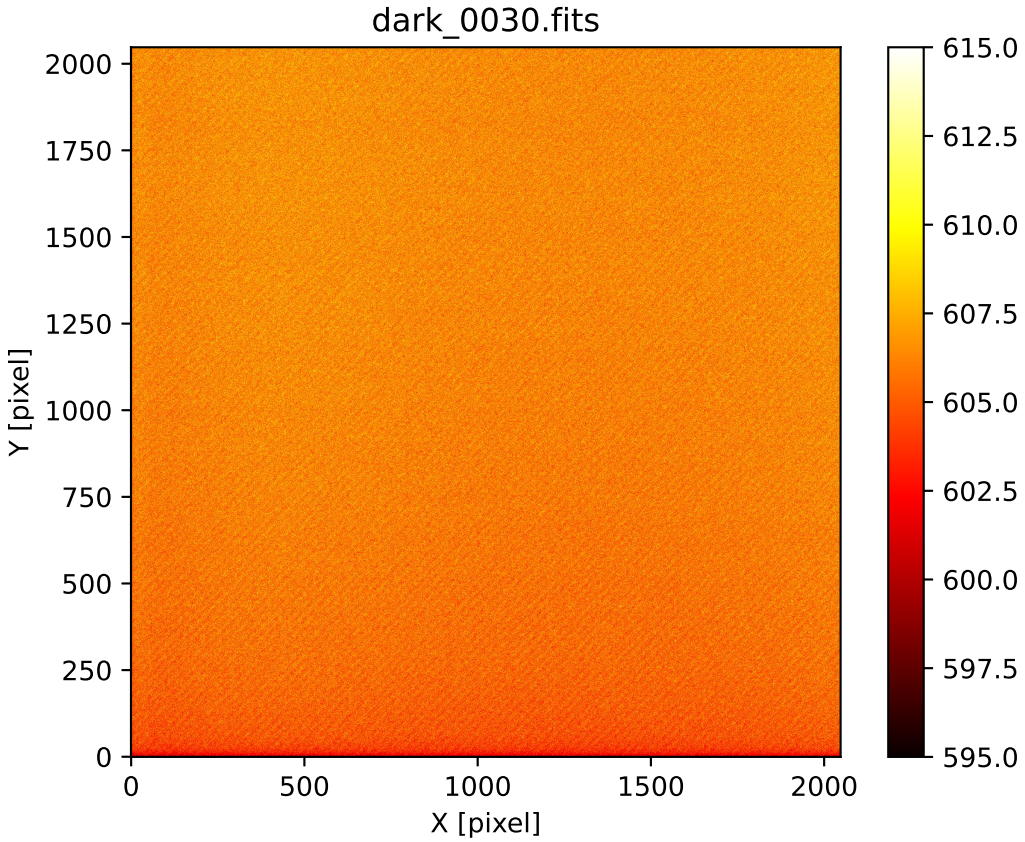

Figure 4: Combined dark frame of 30-sec exposure time.

4.5 Dark subtraction

Make a Python script to subtract one FITS file from another. Here is an example.

Python Code 6: ao2021_s05_06.py

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

# importing argparse module
import argparse
```

```
# importing sys module
import sys

# importing datetime module
import datetime

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Image subtraction'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('minuend', nargs=1, help='minuend (FITS file)')
parser.add_argument ('subtrahend', nargs=1, help='subtrahend (FITS file)')
parser.add_argument ('output', nargs=1, help='output file (FITS file)')

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

# input parameters
file_minuend    = args.minuend[0]
file_subtrahend = args.subtrahend[0]
file_output     = args.output[0]

# command name
command = sys.argv[0]

# printing information
print ("Arithmetic operation being done:")
print ("  %s - %s" % (file_minuend, file_subtrahend) )
print ("    ==> %s" % file_output)

# checking files
for fits in (file_minuend, file_subtrahend, file_output):
    if not (fits[-5:] == '.fits'):
        # printing error message
        print ("Input file must be FITS files!")
        print ("The file \"%s\" does not seem to be a FITS file!" % fits)
        # exit the script
        sys.exit ()

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

# reading minuend FITS file

# opening FITS file
hdu_minuend = astropy.io.fits.open (file_minuend)

# primary HDU
hdu0 = hdu_minuend[0]

# reading header
header = hdu0.header

# reading image
data_minuend = hdu0.data
```

```
# closing FITS file
hdu_minuend.close ()

# reading subtrahend FITS file

# opening FITS file
hdu_subtrahend = astropy.io.fits.open (file_subtrahend)

# primary HDU
hdu1 = hdu_subtrahend[0]

# reading image
data_subtrahend = hdu1.data

# closing FITS file
hdu_subtrahend.close ()

# image subtraction
data_subtracted = data_minuend - data_subtrahend

# adding comments to the header
header['history'] = "FITS file created by the command \"%s\"" % (command)
header['history'] = "Updated on %s" % (now)
header['comment'] = "Image subtraction:"
header['comment'] = "  minuend    = %s" % (file_minuend)
header['comment'] = "  subtrahend = %s" % (file_subtrahend)
header['comment'] = "  output     = %s" % (file_output)

# writing a new FITS file
astropy.io.fits.writeto (file_output, data_subtracted, header=header)
```

Test the script.

```
% chmod a+x ao2021_s05_06.py
% ./ao2021_s05_06.py data_ao2021_s05/lot_20210215_0294.fits dark_0030.fits \
? test_subtraction.fits
Arithmetic operation being done:
  data_ao2021_s05/lot_20210215_0294.fits - dark_0030.fits
    ==> test_subtraction.fits
% ls -l test_subtraction.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:01 test_subtraction.fits
```

Print the header of newly created file.

```
% ./ao2021_s05_05.py test_subtraction.fits
SIMPLE  =                    T / conforms to FITS standard
BITPIX  =                  -64 / array data type
NAXIS   =                    2 / number of array dimensions
NAXIS1  =                 2048
NAXIS2  =                 2048

.....

HISTORY FITS file created by the command "./ao2021_s05_06.py"
HISTORY Updated on 2021-03-16T17:01:46.019192
COMMENT Image subtraction:
```

```
COMMENT   minuend    = data_ao2021_s05/lot_20210215_0294.fits
COMMENT   subtrahend = dark_0030.fits
COMMENT   output     = test_subtraction.fits
```

Check mean pixel values of the raw file and dark subtracted file.

```
% ./ao2021_s05_02.py -d FLAT -e 30 -f gp_Astrodon_2019 \
? data_ao2021_s05/lot_20210215_0294.fits test_subtraction.fits
-------------------------------------------------------------------------------
file name                    n_pix     mean   median    stddev      min      max
===============================================================================
lot_20210215_0294.fits     4194304 13491.83 13503.00    301.88  6889.00 15420.00
test_subtraction.fits      4194304 12885.79 12897.10    301.75  6286.45 14815.05
-------------------------------------------------------------------------------
```

The difference of mean pixel values of raw file and dark subtracted file is about 600 ADU, and the subtraction seems to be OK.

Now, we subtract combined dark frame from raw flatfield frames. We have 20 frames of g'-band dome flatfield. Make a Python script to carry out dark subtraction for those 20 flatfield in convenient way.

Python Code 7: ao2021_s05_07.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.io.fits

# importing subprocess module
import subprocess

# construction of parser object
desc = 'Dark subtraction'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_datatype = ['LIGHT', 'FLAT', 'DARK', 'BIAS']
list_filter = ['gp_Astrodon_2019', 'rp_Astrodon_2019', 'ip_Astrodon_2019', \
               'V_319142', 'R_10349', '__NONE__']
parser.add_argument ('-d', '--datatype', choices=list_datatype, \
                     default='LIGHT', help='accepted data type')
parser.add_argument ('-e', '--exptime', type=float, \
                     default=5.0, help='accepted exposure time')
parser.add_argument ('-f', '--filter', choices=list_filter, \
                     default='__NONE__', help='accepted data type')
parser.add_argument ('-s', '--subtrahend', default='dark.fits', \
                     help='subtrahend FITS file')
parser.add_argument ('files_minuend', nargs='+', help='minuend FITS files')

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

```
# input parameters
files_minuend    = args.files_minuend
file_subtrahend = args.subtrahend
datatype         = args.datatype
exptime          = args.exptime
filter           = args.filter

# processing each file
for file_minuend in files_minuend:
    # if the file is not a FITS file, then skip
    if not (file_minuend[-5:] == '.fits'):
        # printing message
        print ("The file \"%s\" does not seem to be a FITS file!")
        # exit
        sys.exit ()

    # opening FITS file
    hdu_list = astropy.io.fits.open (file_minuend)

    # primary HDU
    hdu0 = hdu_list[0]

    # reading header
    header0 = hdu0.header

    # closing FITS file
    hdu_list.close ()

    # if the FITS file is not what you want, then skip
    if ('FILTER' in header0):
        if not ( (header0['IMAGETYP'] == datatype) \
                 and (header0['EXPTIME'] == exptime) \
                 and (header0['FILTER'] == filter) ):
            continue
    else:
        if not ( (header0['IMAGETYP'] == datatype) \
                 and (header0['EXPTIME'] == exptime) ):
            continue

    # output file name
    file_basename = file_minuend.split ('/')[-1]
    file_output = file_basename[:-5] + '_d.fits'

    # command
    command = "%s %s %s %s" % ("./ao2021_s05_06.py", file_minuend, \
                               file_subtrahend, file_output)

    # printing message
    print ("Now subtracting \"%s\"" % file_subtrahend)
    print ("  from \"%s\"" % file_minuend)
    print ("  and creating \"%s\"" % file_output)

    # executing command
    subprocess.run (command, shell=True)
```

Run the script, and carry out dark subtraction.

```
% chmod a+x ao2021_s05_07.py
```

```
% ./ao2021_s05_07.py -d FLAT -e 30 -f gp_Astrodon_2019 -s dark_0030.fits \
? data_ao2021_s05/*.fits
Now subtracting "dark_0030.fits"
  from "data_ao2021_s05/lot_20210215_0294.fits"
  and creating "lot_20210215_0294_d.fits"
Arithmetic operation being done:
  data_ao2021_s05/lot_20210215_0294.fits - dark_0030.fits
    ==> lot_20210215_0294_d.fits

.....

Now subtracting "dark_0030.fits"
  from "data_ao2021_s05/lot_20210215_0351.fits"
  and creating "lot_20210215_0351_d.fits"
Arithmetic operation being done:
  data_ao2021_s05/lot_20210215_0351.fits - dark_0030.fits
    ==> lot_20210215_0351_d.fits
% ls -l lot_20210215_0*
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0294_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0297_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0300_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0303_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0306_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0309_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0312_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0315_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0318_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0321_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0324_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0327_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0330_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0333_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0336_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0339_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0342_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0345_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0348_d.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 16 17:42 lot_20210215_0351_d.fits
```

Compare the raw file and dark subtracted file.

```
% ./ao2021_s05_02.py -d FLAT -e 30 -f gp_Astrodon_2019 *0294* */*0294*
-------------------------------------------------------------------------------
file name                      n_pix     mean   median   stddev      min      max
===============================================================================
lot_20210215_0294_d.fits   4194304 12885.79 12897.10   301.75  6286.45 14815.05
lot_20210215_0294.fits     4194304 13491.83 13503.00   301.88  6889.00 15420.00
-------------------------------------------------------------------------------
```

The difference of mean pixel values of the raw file and dark subtracted file is about 600 ADU. It seems to be OK.

4.6 Combining dark subtracted flatfield

Combine dark subtracted flatfield frames. Note that the mean level of a raw flatfield frame differs from the mean level of the other raw flatfield frame, and do scaling before combining.

Python Code 8: ao2021_s05_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
import astropy.stats

# importing datetime module
import datetime

# construction of parser object
desc = 'Combining flatfields'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_rejection = ['none', 'sigclip']
list_cenfunc   = ['mean', 'median']
list_datatype  = ['LIGHT', 'FLAT', 'DARK', 'BIAS']
list_filter    = ['gp_Astrodon_2019', 'rp_Astrodon_2019', 'ip_Astrodon_2019', \
                  'V_319142', 'R_10349', '__NONE__']
parser.add_argument ('-d', '--datatype', choices=list_datatype, \
                     default='LIGHT', help='accepted data type')
parser.add_argument ('-e', '--exptime', type=float, \
                     default=5.0, help='accepted exposure time')
parser.add_argument ('-f', '--filter', choices=list_filter, \
                     default='__NONE__', help='accepted data type')
parser.add_argument ('-r', '--rejection', choices=list_rejection, \
                     default='none', help='outlier rejection algorithm')
parser.add_argument ('-t', '--threshold', type=float, default=5.0, \
                     help='rejection threshold in sigma')
parser.add_argument ('-n', '--maxiters', type=int, default=10, \
                     help='maximum number of iterations')
parser.add_argument ('-c', '--cenfunc', choices=list_cenfunc, \
                     default='median', help='method to estimate centre value')
parser.add_argument ('-o', '--output', default='combined.fits', \
                     help='output FITS file')
parser.add_argument ('files', nargs='+', help='input FITS files')

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

# parameters given by command-line arguments
file_input  = args.files
file_output = args.output
rejection   = args.rejection
threshold   = args.threshold
cenfunc     = args.cenfunc
maxiters    = args.maxiters
datatype  = args.datatype
exptime   = args.exptime
```

```
filter     = args.filter

# command name
command = sys.argv[0]

# checking number of intput FITS files
if ( len (file_input) < 2 ):
    # if the number of input files is less than 2, then stop the script
    print ("Number of input files must be 2 or larger!")
    # exit the script
    sys.exit ()

# checking input files
for file_fits in file_input:
    # if the file is not a FITS file, then stop the script
    if not (file_fits[-5:] == '.fits'):
        # printing error message
        print ("Input files must be FITS files!")
        print ("The file \"%s\" is not a FITS file!" % file_fits)
        # exit the script
        sys.exit ()

# checking output file
# if the file is not a FITS file, then stop the script
if not (file_output[-5:] == '.fits'):
    # printing error message
    print ("Output file must be FITS files!")
    # exit the script
    sys.exit ()

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

# parameter for counting
i = 0

# printing information
print ("Data criteria:")
print ("  data type     = %s" % datatype)
print ("  exposure time = %f sec" % exptime)
print ("  filter        = %s" % filter)
print ("List of files to be combined:")

# a list for file names to be combined
file_selected = []

# reading FITS files and constructing a data cube
for file_fits in file_input:
    # opening FITS file
    hdu_list = astropy.io.fits.open (file_fits)

    # primary HDU
    hdu0 = hdu_list[0]

    # reading header
    header0 = hdu0.header

    # if the FITS file is not what you want, then skip
    if ('FILTER' in header0):
```

```
        if not ( (header0['IMAGETYP'] == datatype) \
                 and (header0['EXPTIME'] == exptime) \
                 and (header0['FILTER'] == filter) ):
            # closing FITS file
            hdu_list.close ()
            continue
    else:
        if not ( (header0['IMAGETYP'] == datatype) \
                 and (header0['EXPTIME'] == exptime) ):
            # closing FITS file
            hdu_list.close ()
            continue

    # appending file name to the list "file_selected"
    file_selected.append (file_fits)

    # copying header only for the first FITS file
    if (i == 0):
        header = header0

    # reading data
    data0 = hdu0.data

    # closing FITS file
    hdu_list.close ()

    # calculation of sigma-clipped mean
    mean, median, stddev \
        = astropy.stats.sigma_clipped_stats (data0, sigma=threshold, \
                                             maxiters=maxiters, \
                                             cenfunc=cenfunc, stdfunc='std')

    # normalisation of pixel data
    normalised0 = data0 / mean

    # constructing a data cube
    if (i == 0):
        tmp0 = normalised0
    elif (i == 1):
        cube = numpy.concatenate ( ([tmp0], [normalised0]), axis=0 )
    else:
        cube = numpy.concatenate ( (cube, [normalised0]), axis=0 )

    # incrementing the parameter "i"
    i += 1

    # printing information
    print ("  %s" % file_fits)

# printing information
print ("Output file name: %s" % file_output)
print ("Parameters:")
print ("  rejection = %s" % rejection)
print ("  threshold = %f" % threshold)
print ("  maxiters  = %d" % maxiters)
print ("  cenfunc   = %s" % cenfunc)

# combining images into a single co-added image
if (rejection == 'sigclip'):
```

```
    # combining using sigma clipping
    combined, median, stddev \
        = astropy.stats.sigma_clipped_stats (cube, sigma=threshold, \
                                             maxiters=maxiters, \
                                             cenfunc=cenfunc, stdfunc='std', \
                                             axis=0)
elif (rejection == 'none'):
    # combining using simple mean
    combined = numpy.nanmean (cube, axis=0)

# adding comments to the header
header['history'] = "FITS file created by the command \"%s\"" % (command)
header['history'] = "Updated on %s" % (now)
header['comment'] = "List of combined files:"
for fits in file_selected:
    header['comment'] = "  %s" % (fits)
header['comment'] = "Options given:"
header['comment'] = "  rejection = %s" % (rejection)
header['comment'] = "  threshold = %f sigma" % (threshold)
header['comment'] = "  maxiters  = %d" % (maxiters)
header['comment'] = "  cenfunc   = %s" % (cenfunc)

# writing a new FITS file
astropy.io.fits.writeto (file_output, combined, header=header)
```

Run the script and combine flatfield.

```
% chmod a+x ao2021_s05_08.py
% ./ao2021_s05_08.py -d FLAT -e 30 -f gp_Astrodon_2019 -r sigclip -t 4 \
? -o domeflat_rp.fits lot_20210215_0*.fits
Data criteria:
  data type     = FLAT
  exposure time = 30.000000 sec
  filter        = gp_Astrodon_2019
List of files to be combined:
  lot_20210215_0294_d.fits (mean = 12885.991006)
  lot_20210215_0297_d.fits (mean = 13224.472754)
  lot_20210215_0300_d.fits (mean = 12888.560530)
  lot_20210215_0303_d.fits (mean = 12843.378638)
  lot_20210215_0306_d.fits (mean = 13387.267181)
  lot_20210215_0309_d.fits (mean = 12688.124946)
  lot_20210215_0312_d.fits (mean = 13023.852101)
  lot_20210215_0315_d.fits (mean = 12179.009377)
  lot_20210215_0318_d.fits (mean = 11939.388247)
  lot_20210215_0321_d.fits (mean = 12883.867955)
  lot_20210215_0324_d.fits (mean = 12456.564474)
  lot_20210215_0327_d.fits (mean = 13378.601230)
  lot_20210215_0330_d.fits (mean = 14948.450773)
  lot_20210215_0333_d.fits (mean = 15278.502783)
  lot_20210215_0336_d.fits (mean = 14105.147611)
  lot_20210215_0339_d.fits (mean = 15323.460646)
  lot_20210215_0342_d.fits (mean = 14856.684650)
  lot_20210215_0345_d.fits (mean = 15109.722705)
  lot_20210215_0348_d.fits (mean = 13512.173427)
  lot_20210215_0351_d.fits (mean = 14013.076955)
Output file name: domeflat_rp.fits
Parameters:
  rejection = sigclip
```

```
  threshold = 4.000000
  maxiters  = 10
  cenfunc   = median
% ls -l domeflat_rp.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 16 18:06 domeflat_rp.fits
```

Show the mean value of the combined dome flatfield.

```
% ./ao2021_s05_02.py -d FLAT -e 30 -f gp_Astrodon_2019 domeflat_rp.fits
--------------------------------------------------------------------------------
file name                      n_pix      mean    median    stddev       min       max
================================================================================
domeflat_rp.fits             4194304      1.00      1.00      0.02      0.49      1.07
--------------------------------------------------------------------------------
```

Make a PNG image and display it. (Fig. 5)

```
% ./ao2021_s05_03.py -a 0.85 -b 1.15 -i domeflat_rp.fits -o domeflat_rp.png
domeflat_rp.fits ==> domeflat_rp.png
% feh -dF domeflat_rp.png
```

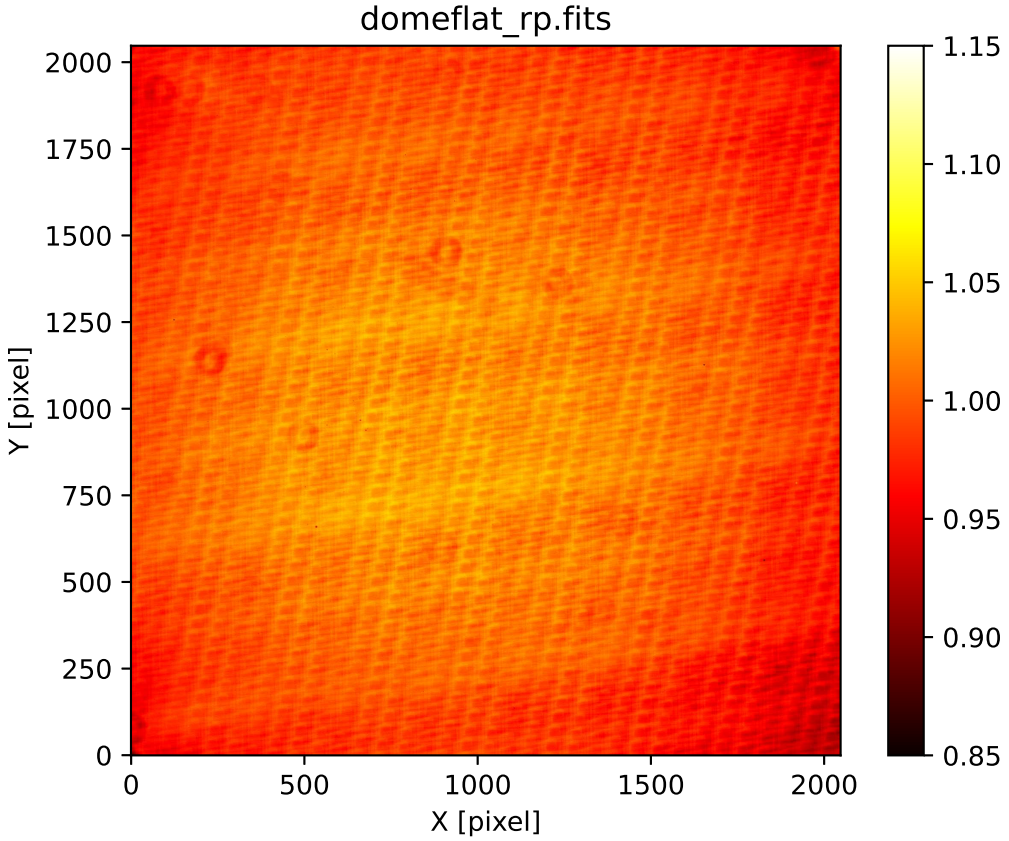

Figure 5: Combined g'-band dome flatfield.

5 Making twilight flatfield

We now construct g'-band twilight flatfield from the data taken on 14/Feb/2021.

5.1 Searching data for dome flatfield

Show g'-band flatfield frames taken on 14/Feb/2021.

```
% ./ao2021_s05_01.py */*20210214*.fits | grep FLAT | grep gp_Astrodon_2019
lot_20210214_0394.fits  2021-02-14  21:35:45  FLAT     45.0  gp_Astrodon_2019
lot_20210214_0399.fits  2021-02-14  21:40:20  FLAT     45.0  gp_Astrodon_2019
lot_20210214_0404.fits  2021-02-14  21:44:55  FLAT     45.0  gp_Astrodon_2019
lot_20210214_0409.fits  2021-02-14  21:49:29  FLAT     45.0  gp_Astrodon_2019
lot_20210214_0414.fits  2021-02-14  21:54:03  FLAT     45.0  gp_Astrodon_2019
lot_20210214_0419.fits  2021-02-14  21:58:38  FLAT     45.0  gp_Astrodon_2019
lot_20210214_0420.fits  2021-02-14  21:59:56  FLAT     15.0  gp_Astrodon_2019
lot_20210214_0425.fits  2021-02-14  22:02:02  FLAT     15.0  gp_Astrodon_2019
lot_20210214_0430.fits  2021-02-14  22:04:25  FLAT      5.0  gp_Astrodon_2019
lot_20210214_0435.fits  2021-02-14  22:05:37  FLAT      5.0  gp_Astrodon_2019
lot_20210214_0440.fits  2021-02-14  22:06:51  FLAT      5.0  gp_Astrodon_2019
lot_20210214_0445.fits  2021-02-14  22:08:03  FLAT      5.0  gp_Astrodon_2019
```

We have found 5-sec, 15-sec, and 45-sec g'-band twilight flatfield frames. We search for 5-sec, 15-sec, and 45-sec dark frames. Here is a list of 5-sec dark frames.

```
% ./ao2021_s05_01.py */*20210214*.fits | grep DARK | grep ' 5.0'
lot_20210214_0552.fits  2021-02-14  22:27:01  DARK      5.0  __NONE__
lot_20210214_0553.fits  2021-02-14  22:27:10  DARK      5.0  __NONE__
lot_20210214_0554.fits  2021-02-14  22:27:20  DARK      5.0  __NONE__
lot_20210214_0555.fits  2021-02-14  22:27:31  DARK      5.0  __NONE__
lot_20210214_0556.fits  2021-02-14  22:27:41  DARK      5.0  __NONE__
lot_20210214_0557.fits  2021-02-14  22:27:51  DARK      5.0  __NONE__
lot_20210214_0558.fits  2021-02-14  22:28:01  DARK      5.0  __NONE__
lot_20210214_0559.fits  2021-02-14  22:28:11  DARK      5.0  __NONE__
lot_20210214_0560.fits  2021-02-14  22:28:21  DARK      5.0  __NONE__
lot_20210214_0561.fits  2021-02-14  22:28:31  DARK      5.0  __NONE__
lot_20210214_0562.fits  2021-02-14  22:28:41  DARK      5.0  __NONE__
lot_20210214_0563.fits  2021-02-14  22:28:51  DARK      5.0  __NONE__
lot_20210214_0564.fits  2021-02-14  22:29:01  DARK      5.0  __NONE__
lot_20210214_0565.fits  2021-02-14  22:29:11  DARK      5.0  __NONE__
lot_20210214_0566.fits  2021-02-14  22:29:21  DARK      5.0  __NONE__
lot_20210214_0567.fits  2021-02-14  22:29:31  DARK      5.0  __NONE__
lot_20210214_0568.fits  2021-02-14  22:29:41  DARK      5.0  __NONE__
lot_20210214_0569.fits  2021-02-14  22:29:51  DARK      5.0  __NONE__
lot_20210214_0570.fits  2021-02-14  22:30:01  DARK      5.0  __NONE__
lot_20210214_0571.fits  2021-02-14  22:30:11  DARK      5.0  __NONE__
```

Following is the list of 15-sec dark frames.

```
% ./ao2021_s05_01.py */*20210214*.fits | grep DARK | grep ' 15.0'
lot_20210214_0632.fits  2021-02-14  22:38:46  DARK     15.0  __NONE__
lot_20210214_0633.fits  2021-02-14  22:39:06  DARK     15.0  __NONE__
lot_20210214_0634.fits  2021-02-14  22:39:26  DARK     15.0  __NONE__
lot_20210214_0635.fits  2021-02-14  22:39:46  DARK     15.0  __NONE__
lot_20210214_0636.fits  2021-02-14  22:40:06  DARK     15.0  __NONE__
lot_20210214_0637.fits  2021-02-14  22:40:26  DARK     15.0  __NONE__
lot_20210214_0638.fits  2021-02-14  22:40:46  DARK     15.0  __NONE__
```

```
lot_20210214_0639.fits  2021-02-14  22:41:06  DARK     15.0  __NONE__
lot_20210214_0640.fits  2021-02-14  22:41:26  DARK     15.0  __NONE__
lot_20210214_0641.fits  2021-02-14  22:41:46  DARK     15.0  __NONE__
lot_20210214_0642.fits  2021-02-14  22:42:06  DARK     15.0  __NONE__
lot_20210214_0643.fits  2021-02-14  22:42:26  DARK     15.0  __NONE__
lot_20210214_0644.fits  2021-02-14  22:42:46  DARK     15.0  __NONE__
lot_20210214_0645.fits  2021-02-14  22:43:06  DARK     15.0  __NONE__
lot_20210214_0646.fits  2021-02-14  22:43:26  DARK     15.0  __NONE__
lot_20210214_0647.fits  2021-02-14  22:43:46  DARK     15.0  __NONE__
lot_20210214_0648.fits  2021-02-14  22:44:06  DARK     15.0  __NONE__
lot_20210214_0649.fits  2021-02-14  22:44:26  DARK     15.0  __NONE__
lot_20210214_0650.fits  2021-02-14  22:44:47  DARK     15.0  __NONE__
lot_20210214_0651.fits  2021-02-14  22:45:07  DARK     15.0  __NONE__
```

And, following is the list of 45-sec dark frames.

```
% ./ao2021_s05_01.py */*20210214*.fits | grep DARK | grep ' 45.0'
lot_20210214_0752.fits  2021-02-14  23:10:34  DARK     45.0  __NONE__
lot_20210214_0753.fits  2021-02-14  23:11:24  DARK     45.0  __NONE__
lot_20210214_0754.fits  2021-02-14  23:12:14  DARK     45.0  __NONE__
lot_20210214_0755.fits  2021-02-14  23:13:04  DARK     45.0  __NONE__
lot_20210214_0756.fits  2021-02-14  23:13:54  DARK     45.0  __NONE__
lot_20210214_0757.fits  2021-02-14  23:14:44  DARK     45.0  __NONE__
lot_20210214_0758.fits  2021-02-14  23:15:34  DARK     45.0  __NONE__
lot_20210214_0759.fits  2021-02-14  23:16:24  DARK     45.0  __NONE__
lot_20210214_0760.fits  2021-02-14  23:17:14  DARK     45.0  __NONE__
lot_20210214_0761.fits  2021-02-14  23:18:04  DARK     45.0  __NONE__
lot_20210214_0762.fits  2021-02-14  23:18:54  DARK     45.0  __NONE__
lot_20210214_0763.fits  2021-02-14  23:19:44  DARK     45.0  __NONE__
lot_20210214_0764.fits  2021-02-14  23:20:34  DARK     45.0  __NONE__
lot_20210214_0765.fits  2021-02-14  23:21:24  DARK     45.0  __NONE__
lot_20210214_0766.fits  2021-02-14  23:22:14  DARK     45.0  __NONE__
lot_20210214_0767.fits  2021-02-14  23:23:04  DARK     45.0  __NONE__
lot_20210214_0768.fits  2021-02-14  23:23:54  DARK     45.0  __NONE__
lot_20210214_0769.fits  2021-02-14  23:24:44  DARK     45.0  __NONE__
lot_20210214_0770.fits  2021-02-14  23:25:34  DARK     45.0  __NONE__
lot_20210214_0771.fits  2021-02-14  23:26:24  DARK     45.0  __NONE__
```

5.2 Examining raw twilight flatfield frames

Show mean pixel values of raw twilight flatfield frames.

Python Code 9: ao2021_s05_09.py

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

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing scipy module
import scipy.stats

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

```
# construction of parser object
desc = 'Calculating statistical values'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_rejection = ['none', 'sigclip']
list_datatype = ['LIGHT', 'FLAT', 'DARK', 'BIAS']
list_filter = ['gp_Astrodon_2019', 'rp_Astrodon_2019', 'ip_Astrodon_2019', \
               'V_319142', 'R_10349', '__NONE__']
parser.add_argument ('-d', '--datatype', choices=list_datatype, \
                     default='LIGHT', help='accepted data type')
parser.add_argument ('-e', '--exptime', type=float, default=5.0, \
                     help='accepted exposure time (negative for any exptime)')
parser.add_argument ('-f', '--filter', choices=list_filter, \
                     default='__NONE__', help='accepted data type')
parser.add_argument ('-r', '--rejection', choices=list_rejection, \
                     default='none', help='outlier rejection algorithm')
parser.add_argument ('-t', '--threshold', type=float, default=4.0, \
                     help='rejection threshold in sigma')
parser.add_argument ('files', nargs='+', help='FITS files')

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

# input parameters
files     = args.files
rejection = args.rejection
threshold = args.threshold
datatype  = args.datatype
exptime   = args.exptime
filter    = args.filter

# printing header
print ("%s" % '-' * 79)
print ("%-25s %8s %8s %8s %8s %8s %8s" \
       % ("file name", "n_pix", "mean", "median", "stddev", "min", "max") )
print ("%s" % '=' * 79)

# processing files
for file in files:
    # if the extension of the file is not '.fits', then skip
    if (file[-5:] != '.fits'):
        continue

    # file name
    path = file.split ('/')
    filename = path[-1]

    # opening FITS file
    hdu_list = astropy.io.fits.open (file)

    # primary HDU
    hdu0 = hdu_list[0]

    # header of primary HDU
    header0 = hdu0.header

    # if the FITS file is not what you want, then skip
```

```
    if ('FILTER' in header0):
        if (exptime < 0.0):
            if not ( (header0['IMAGETYP'] == datatype) \
                     and (header0['FILTER'] == filter) ):
                continue
        else:
            if not ( (header0['IMAGETYP'] == datatype) \
                     and (header0['EXPTIME'] == exptime) \
                     and (header0['FILTER'] == filter) ):
                continue
    else:
        if (exptime < 0.0):
            if not ( (header0['IMAGETYP'] == datatype) ):
                continue
        else:
            if not ( (header0['IMAGETYP'] == datatype) \
                     and (header0['EXPTIME'] == exptime) ):
                continue

    # flattened image data of primary HDU
    data0 = hdu0.data.flatten ()

    # closing FITS file
    hdu_list.close ()

    # if rejection algorithm is used, then do rejection check
    if (rejection == 'sigclip'):
        # sigma clipping using scipy module
        clipped, lower, upper \
            = scipy.stats.sigmaclip (data0, low=threshold, high=threshold)
    elif (rejection == 'none'):
        clipped = data0

    # calculation of statistical values
    n_pix  = len (clipped)
    mean   = numpy.nanmean (clipped)
    median = numpy.nanmedian (clipped)
    stddev = numpy.nanstd (clipped)
    vmin   = numpy.nanmin (clipped)
    vmax   = numpy.nanmax (clipped)

    # printing results
    print ("%-25s %8d %8.2f %8.2f %8.2f %8.2f %8.2f" \
           % (filename, n_pix, mean, median, stddev, vmin, vmax) )

# printing footer
print ("%s" % '-' * 79)
```

Run the script.

```
% chmod a+x ao2021_s05_09.py
% ./ao2021_s05_09.py -d FLAT -e -1 -f gp_Astrodon_2019 -r sigclip -t 5 \
? data_ao2021_s05/*20210214*.fits
-------------------------------------------------------------------------------
file name                   n_pix     mean   median   stddev      min      max
===============================================================================
lot_20210214_0394.fits     4189764   835.00   835.00    19.28   744.00   931.00
lot_20210214_0399.fits     4191160  1144.93  1145.00    30.66   993.00  1298.00
```

```
lot_20210214_0404.fits     4192021  1943.75  1946.00    56.44  1673.00  2225.00
lot_20210214_0409.fits     4192894  3990.62  4000.00   118.86  3405.00  4582.00
lot_20210214_0414.fits     4193536  9357.52  9389.00   280.95  7986.00 10757.00
lot_20210214_0419.fits     4193901 25374.06 25475.00   760.88 21602.00 29155.00
lot_20210214_0420.fits     4193981 11414.75 11455.00   343.55  9748.00 13111.00
lot_20210214_0425.fits     4194073 19723.66 19798.00   589.86 16786.00 22659.00
lot_20210214_0430.fits     4194179 13137.70 13185.00   400.94 11198.00 15134.00
lot_20210214_0435.fits     4194217 18435.41 18509.00   568.38 15732.00 21266.00
lot_20210214_0440.fits     4194231 26100.77 26210.00   809.58 22153.00 30127.00
lot_20210214_0445.fits     4194257 36953.21 37103.00  1126.77 31422.00 42408.00
--------------------------------------------------------------------------------
```

Mean pixel values of raw twilight flatfield frames differs a lot.

5.3 Combining dark frames for twilight flatfield

Combine 5-sec dark frames for twilight flatfield frames.

```
% ./ao2021_s05_04.py -d DARK -e 5 -r sigclip -t 4 -o dark_0005.fits \
? data_ao2021_s05/*20210214*.fits
Data criteria:
  data type     = DARK
  exposure time = 5.000000 sec
  filter        = __NONE__
List of files to be combined:
  data_ao2021_s05/lot_20210214_0552.fits
  data_ao2021_s05/lot_20210214_0553.fits
  data_ao2021_s05/lot_20210214_0554.fits
  data_ao2021_s05/lot_20210214_0555.fits
  data_ao2021_s05/lot_20210214_0556.fits
  data_ao2021_s05/lot_20210214_0557.fits
  data_ao2021_s05/lot_20210214_0558.fits
  data_ao2021_s05/lot_20210214_0559.fits
  data_ao2021_s05/lot_20210214_0560.fits
  data_ao2021_s05/lot_20210214_0561.fits
  data_ao2021_s05/lot_20210214_0562.fits
  data_ao2021_s05/lot_20210214_0563.fits
  data_ao2021_s05/lot_20210214_0564.fits
  data_ao2021_s05/lot_20210214_0565.fits
  data_ao2021_s05/lot_20210214_0566.fits
  data_ao2021_s05/lot_20210214_0567.fits
  data_ao2021_s05/lot_20210214_0568.fits
  data_ao2021_s05/lot_20210214_0569.fits
  data_ao2021_s05/lot_20210214_0570.fits
  data_ao2021_s05/lot_20210214_0571.fits
Output file name: dark_0005.fits
Parameters:
  rejection = sigclip
  threshold = 4.000000
  maxiters  = 10
  cenfunc   = median
% ls -l dark_0005.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 16 20:48 dark_0005.fits
```

Calculate mean pixel value of combined 5-sec dark frame.

```
% ./ao2021_s05_09.py -d DARK -e -1 dark_0005.fits
--------------------------------------------------------------------------------
file name                      n_pix     mean    median    stddev       min       max
================================================================================
dark_0005.fits              4194304   605.62    605.65      1.86    592.55    662.50
--------------------------------------------------------------------------------
```

Similarly, combine 15-sec and 45-sec dark frames.

```
% ./ao2021_s05_04.py -d DARK -e 15 -r sigclip -t 4 -o dark_0015.fits \
? data_ao2021_s05/*20210214*.fits
Data criteria:
  data type     = DARK
  exposure time = 15.000000 sec
  filter        = __NONE__
List of files to be combined:
  data_ao2021_s05/lot_20210214_0632.fits
  data_ao2021_s05/lot_20210214_0633.fits
  data_ao2021_s05/lot_20210214_0634.fits
  data_ao2021_s05/lot_20210214_0635.fits
  data_ao2021_s05/lot_20210214_0636.fits
  data_ao2021_s05/lot_20210214_0637.fits
  data_ao2021_s05/lot_20210214_0638.fits
  data_ao2021_s05/lot_20210214_0639.fits
  data_ao2021_s05/lot_20210214_0640.fits
  data_ao2021_s05/lot_20210214_0641.fits
  data_ao2021_s05/lot_20210214_0642.fits
  data_ao2021_s05/lot_20210214_0643.fits
  data_ao2021_s05/lot_20210214_0644.fits
  data_ao2021_s05/lot_20210214_0645.fits
  data_ao2021_s05/lot_20210214_0646.fits
  data_ao2021_s05/lot_20210214_0647.fits
  data_ao2021_s05/lot_20210214_0648.fits
  data_ao2021_s05/lot_20210214_0649.fits
  data_ao2021_s05/lot_20210214_0650.fits
  data_ao2021_s05/lot_20210214_0651.fits
Output file name: dark_0015.fits
Parameters:
  rejection = sigclip
  threshold = 4.000000
  maxiters  = 10
  cenfunc   = median
% ./ao2021_s05_04.py -d DARK -e 45 -r sigclip -t 4 -o dark_0045.fits \
? data_ao2021_s05/*20210214*.fits
Data criteria:
  data type     = DARK
  exposure time = 45.000000 sec
  filter        = __NONE__
List of files to be combined:
  data_ao2021_s05/lot_20210214_0752.fits
  data_ao2021_s05/lot_20210214_0753.fits
  data_ao2021_s05/lot_20210214_0754.fits
  data_ao2021_s05/lot_20210214_0755.fits
  data_ao2021_s05/lot_20210214_0756.fits
  data_ao2021_s05/lot_20210214_0757.fits
  data_ao2021_s05/lot_20210214_0758.fits
  data_ao2021_s05/lot_20210214_0759.fits
  data_ao2021_s05/lot_20210214_0760.fits
```

```
  data_ao2021_s05/lot_20210214_0761.fits
  data_ao2021_s05/lot_20210214_0762.fits
  data_ao2021_s05/lot_20210214_0763.fits
  data_ao2021_s05/lot_20210214_0764.fits
  data_ao2021_s05/lot_20210214_0765.fits
  data_ao2021_s05/lot_20210214_0766.fits
  data_ao2021_s05/lot_20210214_0767.fits
  data_ao2021_s05/lot_20210214_0768.fits
  data_ao2021_s05/lot_20210214_0769.fits
  data_ao2021_s05/lot_20210214_0770.fits
  data_ao2021_s05/lot_20210214_0771.fits
Output file name: dark_0045.fits
Parameters:
  rejection = sigclip
  threshold = 4.000000
  maxiters  = 10
  cenfunc   = median
% ls -l dark_00??.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 16 20:48 dark_0005.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 16 21:03 dark_0015.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 16 16:15 dark_0030.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 16 21:04 dark_0045.fits
```

5.4 Subtracting dark frame from raw flatfield frames

Make a Python script to subtract dark from raw flatfield frames.

Python Code 10: ao2021_s05_10.py

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

# importing argparse
import argparse

# importing astropy module
import astropy.io.fits

# importing subprocess module
import subprocess

# construction of parser object
desc = 'Subtracting dark from raw flatfield'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_filter = ['gp_Astrodon_2019', 'rp_Astrodon_2019', 'ip_Astrodon_2019', \
               'V_319142', 'R_10349', '__NONE__']
parser.add_argument ('-f', '--filter', choices=list_filter, \
                     default='__NONE__', help='filter')
parser.add_argument ('files', nargs='+', help='FITS files')

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

# input parameters
files  = args.files
filter = args.filter

# processing files one-by-one
```

```
for file_fits in files:
    # if the extension of the file is not '.fits', then skip
    if (file_fits[-5:] != '.fits'):
        continue

    # opening FITS file
    hdu_list = astropy.io.fits.open (file_fits)

    # primary HDU
    hdu0 = hdu_list[0]

    # header of primary HDU
    header0 = hdu0.header

    # closing FITS file
    hdu_list.close ()

    # skip, if the data type is not 'FLAT'
    if not (header0['IMAGETYP'] == 'FLAT'):
        continue

    # skip, if filter is not the one specified by the command-line argument
    if not (header0['FILTER'] == filter):
        continue

    # exposure time
    exptime = float (header0['EXPTIME'])

    # combined dark frame file name
    file_dark = "dark_%04d.fits" % exptime

    # output FITS file name
    file_output = file_fits.split ('/')[-1][:-5] + '_d.fits'

    # dark subtraction command
    command = "%s %s %s %s" % ('./ao2021_s05_06.py', file_fits, \
                               file_dark, file_output)

    # printing message
    print ("Now subtracting \"%s\"" % file_fits)
    print ("  from \"%s\"" % file_dark)
    print ("  and creating \"%s\"" % file_output)

    # executing command
    subprocess.run (command, shell=True)
```

Execute the script, and carry out dark subtraction.

```
% chmod a+x ao2021_s05_10.py
% ./ao2021_s05_10.py -f gp_Astrodon_2019 data_ao2021_s05/*20210214*.fits
Now subtracting "data_ao2021_s05/lot_20210214_0394.fits"
  from "dark_0045.fits"
  and creating "lot_20210214_0394_d.fits"
Arithmetic operation being done:
  data_ao2021_s05/lot_20210214_0394.fits - dark_0045.fits
    ==> lot_20210214_0394_d.fits

.....
```

```
Now subtracting "data_ao2021_s05/lot_20210214_0445.fits"
  from "dark_0005.fits"
  and creating "lot_20210214_0445_d.fits"
Arithmetic operation being done:
  data_ao2021_s05/lot_20210214_0445.fits - dark_0005.fits
    ==> lot_20210214_0445_d.fits
```

Compare raw flatfield and dark subtracted flatfield frames.

```
% ./ao2021_s05_09.py -d FLAT -e -1 -f gp_Astrodon_2019 *440*.fits */*440*.fits
--------------------------------------------------------------------------------
file name                      n_pix     mean   median   stddev      min      max
================================================================================
lot_20210214_0440_d.fits     4194304 25495.23 25604.35   810.41 12235.35 46981.35
lot_20210214_0440.fits       4194304 26100.85 26210.00   810.51 12844.00 47590.00
--------------------------------------------------------------------------------
```

The difference of mean pixel values of raw and dark subtracted flatfield frames are about 600 ADU, and it seems to be OK.

5.5 Combining dark subtracted flatfield frames

Make a Python script to combine dark subtracted flatfield frames. Note that mean values of twilight flatfield frames differ a lot and weighted average is useful to have a better result.

Python Code 11: ao2021_s05_11.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing numpy module
import numpy
import numpy.ma

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

# importing datetime module
import datetime

# construction of parser object
desc = 'Combining twilight flatfields'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_rejection = ['none', 'sigclip']
list_cenfunc   = ['mean', 'median']
list_filter    = ['gp_Astrodon_2019', 'rp_Astrodon_2019', 'ip_Astrodon_2019', \
                  'V_319142', 'R_10349', '__NONE__']
parser.add_argument ('-f', '--filter', choices=list_filter, \
                     default='__NONE__', help='accepted data type')
```

```
parser.add_argument ('-r', '--rejection', choices=list_rejection, \
                     default='none', help='outlier rejection algorithm')
parser.add_argument ('-t', '--threshold', type=float, default=5.0, \
                     help='rejection threshold in sigma')
parser.add_argument ('-n', '--maxiters', type=int, default=10, \
                     help='maximum number of iterations')
parser.add_argument ('-c', '--cenfunc', choices=list_cenfunc, \
                     default='median', help='method to estimate centre value')
parser.add_argument ('-m', '--max', type=float, \
                     default=30000.0, help='maximum mean value for use')
parser.add_argument ('-o', '--output', default='combined.fits', \
                     help='output FITS file')
parser.add_argument ('files', nargs='+', help='input FITS files')

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

# parameters given by command-line arguments
file_input  = args.files
file_output = args.output
rejection   = args.rejection
threshold   = args.threshold
cenfunc     = args.cenfunc
maxiters    = args.maxiters
filter      = args.filter
limit_max   = args.max

# command name
command = sys.argv[0]

# checking number of intput FITS files
if ( len (file_input) < 2 ):
    # if the number of input files is less than 2, then stop the script
    print ("Number of input files must be 2 or larger!")
    # exit the script
    sys.exit ()

# checking input files
for file_fits in file_input:
    # if the file is not a FITS file, then stop the script
    if not (file_fits[-5:] == '.fits'):
        # printing error message
        print ("Input files must be FITS files!")
        print ("The file \"%s\" is not a FITS file!" % file_fits)
        # exit the script
        sys.exit ()

# checking output file
# if the file is not a FITS file, then stop the script
if not (file_output[-5:] == '.fits'):
    # printing error message
    print ("Output file must be FITS files!")
    # exit the script
    sys.exit ()

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

# parameter for counting
```

```
i = 0

# printing information
print ("Data criteria:")
print ("  filter        = %s" % filter)
print ("List of files to be combined:")

# a list for file names to be combined
file_selected = []

# a Numpy array for weighted average
weight = numpy.array ([], dtype='float64')

# reading FITS files and constructing a data cube
for file_fits in file_input:
    # opening FITS file
    hdu_list = astropy.io.fits.open (file_fits)

    # primary HDU
    hdu0 = hdu_list[0]

    # reading header
    header0 = hdu0.header

    # if the FITS file is not flatfield, then skip
    if not (header0['IMAGETYP'] == 'FLAT'):
        # closing FITS file
        hdu_list.close ()
        continue

    # if it is not an image taken by filter of your interest, then skip
    if not (header0['FILTER'] == filter):
        # closing FITS file
        hdu_list.close ()
        continue

    # copying header only for the first FITS file
    if (i == 0):
        header = header0

    # reading data
    data0 = hdu0.data

    # closing FITS file
    hdu_list.close ()

    # calculation of sigma-clipped mean
    mean, median, stddev \
        = astropy.stats.sigma_clipped_stats (data0, sigma=threshold, \
                                             maxiters=maxiters, \
                                             cenfunc=cenfunc, stdfunc='std')

    # if mean value is greater than "limit_max", then skip
    if (mean > limit_max):
        continue

    # appending file name to the list "file_selected"
    file_selected.append (file_fits)
```

```
    # appending mean value to the array "weight"
    weight = numpy.append (weight, mean)

    # normalisation of pixel data
    normalised0 = data0 / mean

    # constructing a data cube
    if (i == 0):
        tmp0 = normalised0
    elif (i == 1):
        cube = numpy.concatenate ( ([tmp0], [normalised0]), axis=0 )
    else:
        cube = numpy.concatenate ( (cube, [normalised0]), axis=0 )

    # incrementing the parameter "i"
    i += 1

    # printing information
    print ("  %s (mean = %f)" % (file_fits, mean) )

# printing information
print ("Output file name: %s" % file_output)
print ("Parameters:")
print ("  rejection = %s" % rejection)
print ("  threshold = %f" % threshold)
print ("  maxiters  = %d" % maxiters)
print ("  cenfunc   = %s" % cenfunc)

# combining images into a single co-added image
if (rejection == 'sigclip'):
    # sigma clipping
    cube_clipped = astropy.stats.sigma_clip (cube, sigma=threshold, \
                                             maxiters=maxiters, \
                                             cenfunc=cenfunc, stdfunc='std', \
                                             axis=0, masked=True)
    # weighted average
    combined = numpy.ma.average (cube_clipped, weights=weight, axis=0)
elif (rejection == 'none'):
    # weighted average
    combined = numpy.average (cube, weights=weight, axis=0)

# adding comments to the header
header['history'] = "FITS file created by the command \"%s\"" % (command)
header['history'] = "Updated on %s" % (now)
header['comment'] = "Weighted average is used for combining"
header['comment'] = "List of combined files:"
for fits in file_selected:
    header['comment'] = "  %s" % (fits)
header['comment'] = "Options given:"
header['comment'] = "  rejection = %s" % (rejection)
header['comment'] = "  threshold = %f sigma" % (threshold)
header['comment'] = "  maxiters  = %d" % (maxiters)
header['comment'] = "  cenfunc   = %s" % (cenfunc)

# writing a new FITS file
astropy.io.fits.writeto (file_output, \
                         numpy.ma.filled (combined, fill_value=numpy.nan), \
                         header=header)
```

Execute the script, and make a flatfield.

```
% chmod a+x ao2021_s05_11.py
% ./ao2021_s05_11.py -f gp_Astrodon_2019 -r sigclip -t 2.5 -o twiflat_rp.fits \
? lot_20210214_0*.fits
Data criteria:
  filter        = gp_Astrodon_2019
List of files to be combined:
  lot_20210214_0394_d.fits (mean = 229.337764)
  lot_20210214_0399_d.fits (mean = 539.517249)
  lot_20210214_0404_d.fits (mean = 1339.507473)
  lot_20210214_0409_d.fits (mean = 3390.631761)
  lot_20210214_0414_d.fits (mean = 8771.484739)
  lot_20210214_0419_d.fits (mean = 24830.706824)
  lot_20210214_0420_d.fits (mean = 10834.176364)
  lot_20210214_0425_d.fits (mean = 19162.455028)
  lot_20210214_0430_d.fits (mean = 12562.283792)
  lot_20210214_0435_d.fits (mean = 17878.652957)
  lot_20210214_0440_d.fits (mean = 25571.347555)
Output file name: twiflat_rp.fits
Parameters:
  rejection = sigclip
  threshold = 2.500000
  maxiters  = 10
  cenfunc   = median
% ls -l twiflat_rp.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 17 00:21 twiflat_rp.fits
```

Check the mean pixel value of combined flatfield.

```
% ./ao2021_s05_09.py -d FLAT -e -1 -f gp_Astrodon_2019 twiflat_rp.fits
--------------------------------------------------------------------------------
file name                      n_pix     mean   median   stddev      min      max
================================================================================
twiflat_rp.fits             4194304     1.00     1.00     0.03     0.48     1.11
--------------------------------------------------------------------------------
```

Make a PNG file and show it on the display. (Fig. 6)

```
% ./ao2021_s05_03.py -a 0.85 -b 1.15 -i twiflat_rp.fits -o twiflat_rp.png
twiflat_rp.fits ==> twiflat_rp.png
% feh -dF twiflat_rp.png
```

6 For your training

- Read chapter 4 of "Handbook of CCD Astronomy" and learn about flatfield frame.
  - Handbook of CCD Astronomy (2nd Edition)
    - Steve B. Howell
    - Cambridge University Press
    - https://doi.org/10.1017/CBO9780511807909
- Visit the official website of Numpy, and learn about masked arrays.
  - https://numpy.org/
  - https://numpy.org/doc/stable/reference/maskedarray.html

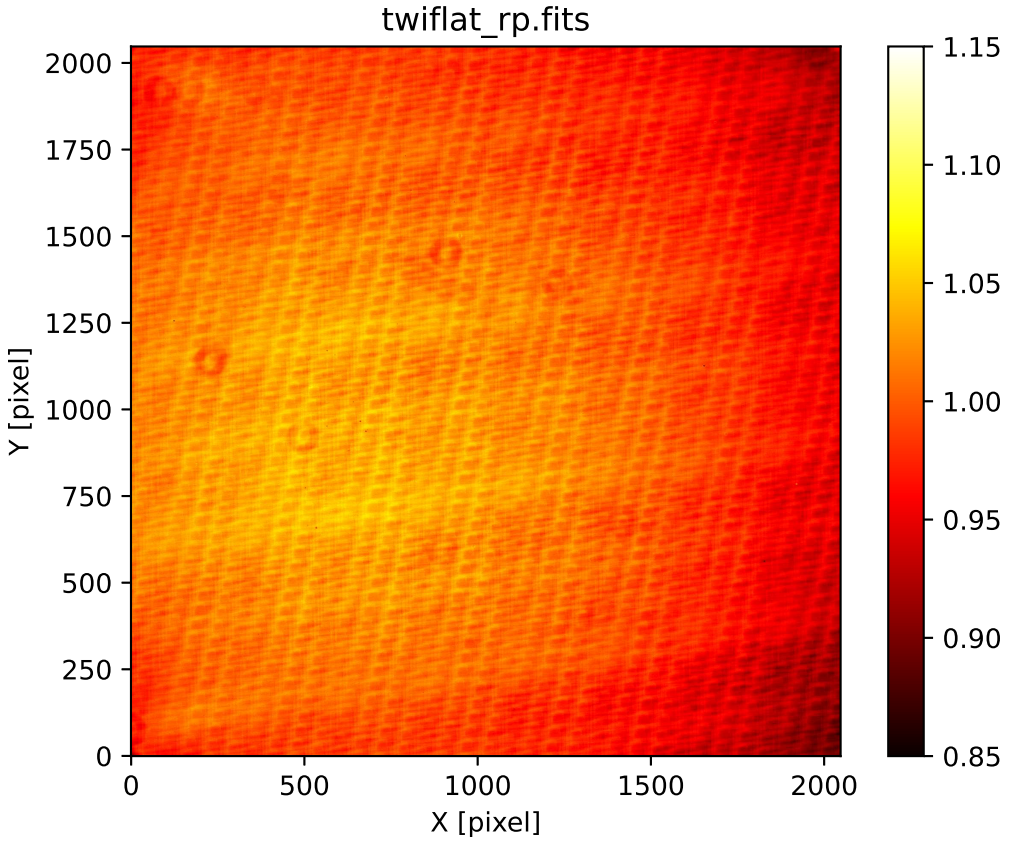

Figure 6: The g'-band combined twilight flatfield.

7 Assignment

1. What is flatfield? Describe flatfield. List three different types of flatfield, and describe each type of flatfield. How to take flatfield frames? Why do we need flatfield?

2. Choose three raw flatfield frames. Make your own Python script to visualise those three raw flatfield frames using Matplotlib. Show the source code of your Python script and images you have produced.

3. Choose three raw flatfield frames. Make your own Python script to make histograms of those three raw flatfield frames using Matplotlib. Show the source code of your Python script and histograms you have produced. Explain how you decide the width of the bin for histogram.

4. Choose three raw flatfield frames. Make your own Python script to calculate mean, median, variance, standard deviation, minimum value, maximum value of those three raw flatfield frames without using convenient functions of Numpy (e.g. numpy.mean, numpy.std, etc.). Show the source code of your Python script. Take a screenshot of your computer display after the execution of your Python script, and show the result of your calculation.

5. Choose three raw flatfield frames. Make your own Python script to calculate mean, median, variance, standard deviation, minimum value, maximum value of those three raw flatfield frames using convenient functions of Numpy (e.g. numpy.mean, numpy.std, etc.). Show the source code of your Python script. Take a screenshot of your computer display after the execution of your Python script, and show the result of your calculation.

6. Make your own Python script which works like `imarith` task of IRAF (Image Reduction and Analysis Facility). Show the source code of your script. Test your script with some dark data for this session. Take a screenshot of your computer display to show the results. Mention which functions of `imarith` are implemented in your Python script. Mention which functions of `imarith` are not implemented in your Python script. About `imarith` task, you may refer to the IRAF's help.

   - `https://iraf.net/irafhelp.php?val=imarith`

7. Make your own Python script to costruct r'-band and i'-band dome flatfield using the data obtained on 15/Feb/2021. Show the source code of your Python script. Execute the script, and show generated r'-band dome flatfield. Describe the method of making combined dome flatfield from a set of raw flatfield frames.

8. Make your own Python script to construct r'-band and i'-band twilight flatfield using the data obtained on 14/Feb/2021. Show the source code of your Python script. Execute the script, and show generated r'-band dome flatfield. Describe the method of making combined twilight flatfield from a set of raw flatfield frames.

9. What is weighted average? Describe weighted average. How to calculate weighted average? How useful is it? Under which condition, weighted average should be used?

10. Make a Python script to calculate weighted average. Show the source code of your script. Execute the script, and take a screenshot of your computer display to show the results.

11. Describe masked arrays of Numpy. Show some possible applications of Numpy's masked arrays for analysis of astronomical observational data.

12. Make a Python script to utilise masked arrays of Numpy. Show the source code of your script. Execute the script, and take a screenshot of your computer display to show the results.