Advanced Astronomical Observations 2021
Session 04: Dark Current

Kinoshita Daisuke

10 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 examine dark frames. Bias frames are images of non-zero second exposure with shutter closed.

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 350 MB.

- `https://s3b.astro.ncu.edu.tw/advobs_202102/data/data_ao2021_s04.tar.xz`

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

```
% curl -k -o data_ao2021_s04.tar.xz \
? https://s3b.astro.ncu.edu.tw/advobs_202102/data/data_ao2021_s04.tar.xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  353M  100  353M    0     0  5525k      0  0:01:05  0:01:05 --:--:-- 5551k
```

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. 130 FITS files should be extracted from the archive file.

```
% tar xJvf data_ao2021_s04.tar.xz
x data_ao2021_s04/
x data_ao2021_s04/lot_20210210_1020.fits
x data_ao2021_s04/lot_20210210_1021.fits
x data_ao2021_s04/lot_20210210_1022.fits
x data_ao2021_s04/lot_20210210_1023.fits
x data_ao2021_s04/lot_20210210_1024.fits

.....

x data_ao2021_s04/lot_20210210_1145.fits
x data_ao2021_s04/lot_20210210_1146.fits
x data_ao2021_s04/lot_20210210_1147.fits
x data_ao2021_s04/lot_20210210_1148.fits
x data_ao2021_s04/lot_20210210_1149.fits
% ls data_ao2021_s04/*.fits | wc
     130     130    5070
```

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

```
% unxz -c data_ao2021_s04.tar.xz | tar xvf -
x data_ao2021_s04/
x data_ao2021_s04/lot_20210210_1020.fits
x data_ao2021_s04/lot_20210210_1021.fits
x data_ao2021_s04/lot_20210210_1022.fits
x data_ao2021_s04/lot_20210210_1023.fits
x data_ao2021_s04/lot_20210210_1024.fits

.....

x data_ao2021_s04/lot_20210210_1145.fits
x data_ao2021_s04/lot_20210210_1146.fits
x data_ao2021_s04/lot_20210210_1147.fits
x data_ao2021_s04/lot_20210210_1148.fits
x data_ao2021_s04/lot_20210210_1149.fits
% ls data_ao2021_s04/*.fits | wc
     130     130    5070
```

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_s04.tar.xz
-rw-r--r--  1 daisuke  taiwan  370696560 Mar  9 15:59 data_ao2021_s04.tar.xz
% unxz data_ao2021_s04.tar.xz
% ls -l data_ao2021_s04.tar
-rw-r--r--  1 daisuke  taiwan  1091452416 Mar  9 15:59 data_ao2021_s04.tar
% tar xvf data_ao2021_s04.tar
x data_ao2021_s04/
x data_ao2021_s04/lot_20210210_1020.fits
x data_ao2021_s04/lot_20210210_1021.fits
x data_ao2021_s04/lot_20210210_1022.fits
x data_ao2021_s04/lot_20210210_1023.fits
x data_ao2021_s04/lot_20210210_1024.fits
```

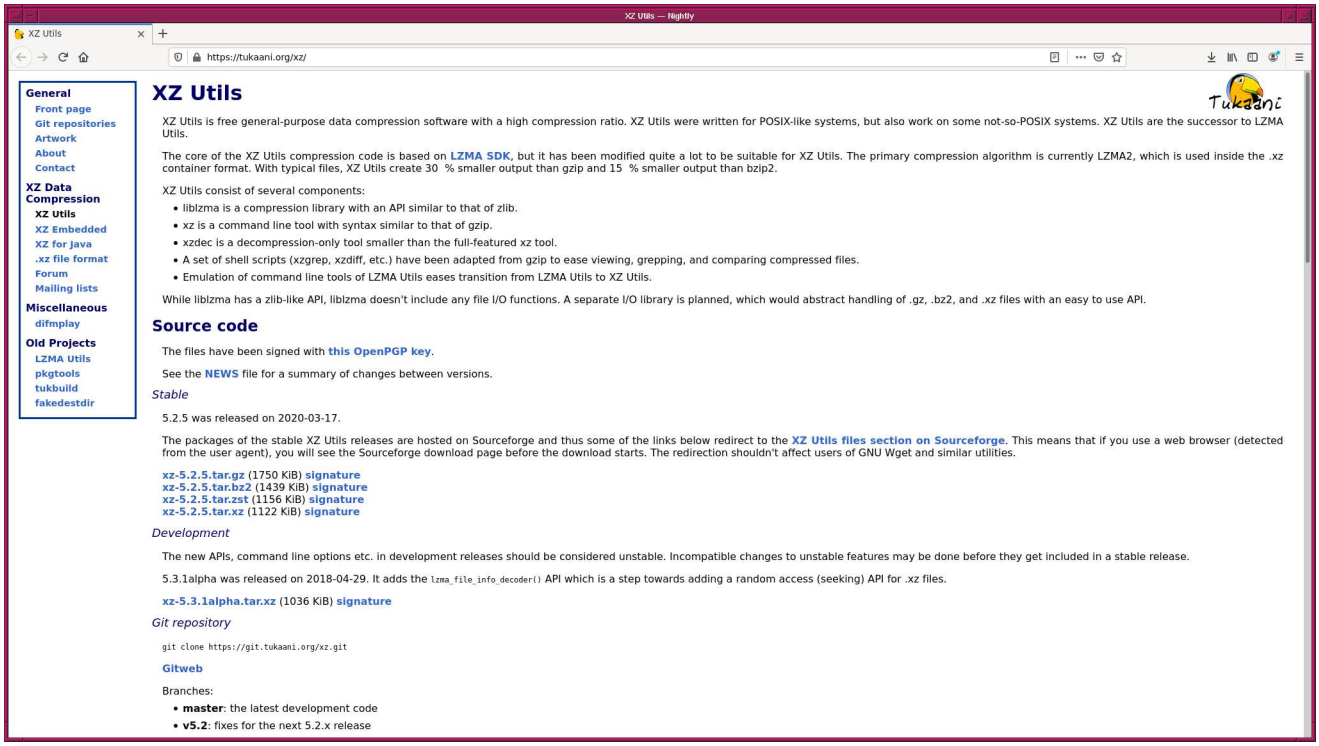

Figure 1: The website of XZ Utils.

```
.....

x data_ao2021_s04/lot_20210210_1145.fits
x data_ao2021_s04/lot_20210210_1146.fits
x data_ao2021_s04/lot_20210210_1147.fits
x data_ao2021_s04/lot_20210210_1148.fits
x data_ao2021_s04/lot_20210210_1149.fits
% ls data_ao2021_s04/*.fits | wc
     130     130    5070
```

3 Dealing with 3-dimensional Numpy arrays

For this session, the knowledge of manipulation of multi-dimensional Numpy arrays is required. Before handling dark frames, we try some Python scripts using Numpy.

3.1 Checking whether you have Numpy on your computer

Try following to check whether you have Numpy on your computer.

```
% python3.9
Python 3.9.2 (default, Feb 21 2021, 12:39:42)
[GCC 7.5.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> exit ()
```

If you successfully import Numpy without any error message, you have Numpy properly installed on your computer. If you do not have Numpy on your computer, then you probably see a message like following.

```
% python3.9
Python 3.9.1 (default, Jan 24 2021, 12:55:46)
[GCC 7.5.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'numpy'
>>>
```

If you do not have Numpy on your computer, then visit the official website of Numpy (Fig. 2). Read the official documentation and install Numpy on your computer.

- https://numpy.org/

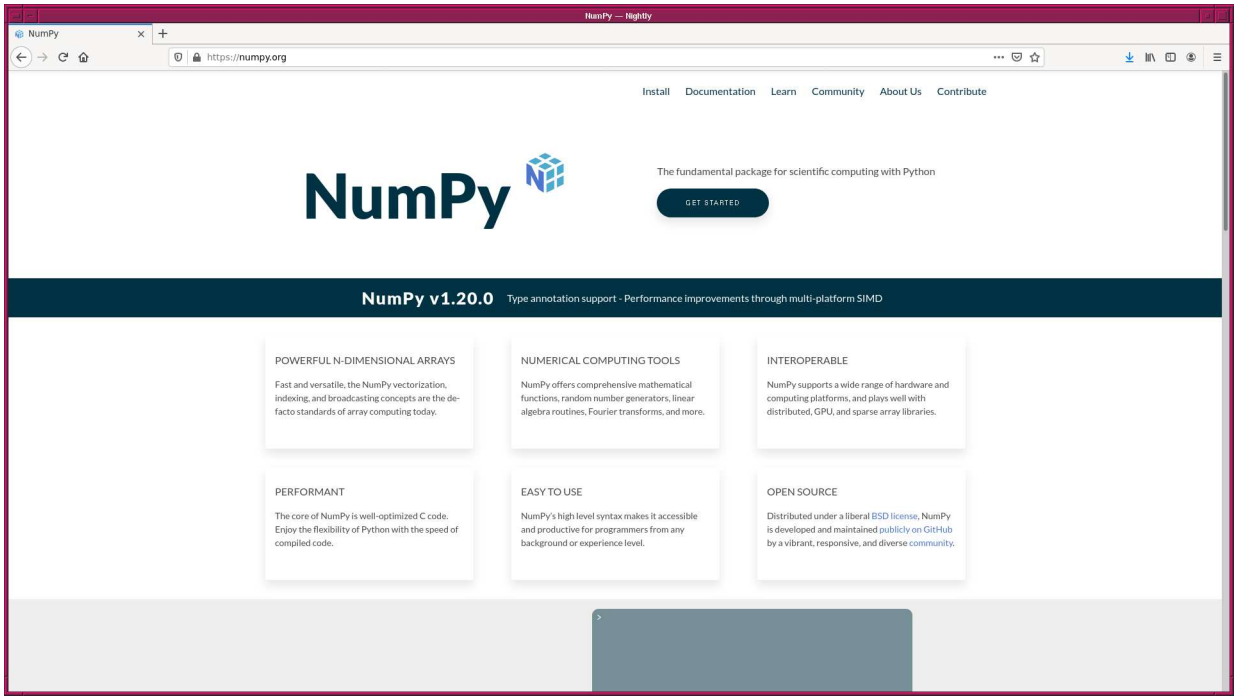

Figure 2: The official website of Numpy.

3.2 Creating a Numpy array

Make a Python script to create a Numpy array.

Python Code 1: ao2021_s04_01.py

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

# importing Numpy module
import numpy

# creating a Numpy array
a = numpy.array ([1.2, 3.4, 5.6, 7.8, 9.0, 12.3], dtype='float64')

# printing Numpy array
print (a)
```

Execute the script.

```
% chmod a+x ao2021_s04_01.py
% ./ao2021_s04_01.py
[ 1.2  3.4  5.6  7.8  9.  12.3]
```

3.3 Creating a 2-dim. Numpy array

Make a Python script to create a 2-dimensional Numpy array.

Python Code 2: ao2021_s04_02.py

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

# importing Numpy module
import numpy

# creating a Numpy array
b = numpy.array ([ [1.2, 3.4, 5.6], [7.8, 9.0, 12.3], [1.0, 2.0, 3.0] ], \
                 dtype='float64')

# printing Numpy array
print (b)
```

Execute the script.

```
% chmod a+x ao2021_s04_02.py
% ./ao2021_s04_02.py
[[ 1.2  3.4  5.6]
 [ 7.8  9.  12.3]
 [ 1.   2.   3. ]]
```

3.4 Arithmetic operations of 2-dim. Numpy arrays

Make a Python script to carry out arithmetic operations of 2-dimensional Numpy arrays.

Python Code 3: ao2021_s04_03.py

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

# importing Numpy module
import numpy

# creating Numpy arrays
a = numpy.array ([ [10.0, 11.0, 12.0], \
                   [13.0, 14.0, 15.0], \
                   [16.0, 17.0, 18.0] ], \
                 dtype='float64')
b = numpy.array ([ [1.0, 2.0, 3.0], \
                   [4.0, 5.0, 6.0], \
                   [7.0, 8.0, 9.0] ], \
                 dtype='float64')
c = numpy.array ([ [1.0, 1.1, 0.9], \
                   [1.2, 0.8, 1.3], \
                   [0.7, 1.4, 0.6] ], \
                 dtype='float64')
```

```
# arithmetic operations
d = a - b
e = a / c

# printing Numpy array
print ("a:")
print (a)
print ("b:")
print (b)
print ("c:")
print (c)
print ("d = a - b:")
print (d)
print ("e = a / c:")
print (e)
```

Execute the script.

```
% chmod 755 ao2021_s04_03.py
% a:
[[10. 11. 12.]
 [13. 14. 15.]
 [16. 17. 18.]]
b:
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
c:
[[1.  1.1 0.9]
 [1.2 0.8 1.3]
 [0.7 1.4 0.6]]
d = a - b:
[[9. 9. 9.]
 [9. 9. 9.]
 [9. 9. 9.]]
e = a / c:
[[10.         10.         13.33333333]
 [10.83333333 17.5        11.53846154]
 [22.85714286 12.14285714 30.        ]]
```

3.5 Creating a 3-dim. array from a set of 2-dim. arrays

Make a Python script to create a 3-dimensional array from a set of 2-dimensional arrays.

Python Code 4: ao2021_s04_04.py

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

# importing Numpy module
import numpy
import numpy.random

# parameters
n_x = 5
n_y = 5
mean  = 50.0
sigma = 10.0
```

```
# creating Numpy arrays
a = numpy.random.normal (mean, sigma, (n_x, n_y))
b = numpy.random.normal (mean, sigma, (n_x, n_y))
c = numpy.random.normal (mean, sigma, (n_x, n_y))
d = numpy.random.normal (mean, sigma, (n_x, n_y))
e = numpy.random.normal (mean, sigma, (n_x, n_y))

# printing information of Numpy arrays
print ("shape of a =", a.shape)
print ("shape of b =", b.shape)
print ("shape of c =", c.shape)
print ("shape of d =", d.shape)
print ("shape of e =", e.shape)

# concatenating arrays
cube = numpy.concatenate ( ([a], [b]), axis=0 )

# printing information of Numpy array "cube"
print ("shape of cube =", cube.shape)

# concatenating one more array
cube = numpy.concatenate ( (cube, [c]), axis=0 )

# printing information of Numpy array "cube"
print ("shape of cube =", cube.shape)

# concatenating 5 arrays
cube2 = numpy.concatenate ( ([a], [b], [c], [d], [e]), axis=0 )

# printing information of Numpy array "cube"
print ("shape of cube2 =", cube2.shape)

# printing cube2
print ("cube2 =")
print (cube2)
```

Execute the script.

```
% chmod a+x ao2021_s04_04.py
% ./ao2021_s04_04.py
shape of a = (5, 5)
shape of b = (5, 5)
shape of c = (5, 5)
shape of d = (5, 5)
shape of e = (5, 5)
shape of cube = (2, 5, 5)
shape of cube = (3, 5, 5)
shape of cube2 = (5, 5, 5)
cube2 =
[[[52.37942245 36.23775475 59.53156359 43.38459542 59.69198563]
  [58.82438873 40.0150905  57.79518356 56.31328182 59.42072224]
  [47.03294562 43.25407159 47.57578236 55.01158043 31.79988512]
  [37.712273   63.44526052 61.488259   56.33718833 51.96745718]
  [40.22258049 41.6359386  53.81574953 54.91882859 46.30350838]]

 [[49.00244066 44.80618249 54.7952967  44.60739012 42.5625902 ]
  [32.50846763 48.3822386  64.25240712 55.15165086 72.71057684]
  [54.38181346 38.26123861 31.15585387 38.4417197  48.82454833]
```

```
  [45.14764536 50.72518264 46.91975383 38.37557793 52.14946044]
  [45.54007492 54.60721581 27.08384773 52.08292726 54.25305728]]

 [[61.51202131 56.13628775 41.53808764 46.90685425 74.61364523]
  [52.92484356 61.40586975 38.32239967 53.36192577 49.93916544]
  [54.67236664 57.77251115 56.29918875 42.35195716 67.92081057]
  [55.04149642 41.60233567 47.27292034 65.35725972 61.25016538]
  [40.03115727 51.52575143 40.71343093 64.61643576 65.00399879]]

 [[34.1160526  56.20827672 53.51504724 31.92101388 57.2901461 ]
  [42.92803059 66.13400113 68.97966226 57.96660404 43.98008816]
  [44.80593241 56.58165665 34.73949989 42.9612481  45.61519716]
  [44.04953208 44.02725116 44.34768962 61.51075395 56.58309649]
  [56.33513844 52.22749709 56.05259066 52.67011809 51.08730104]]

 [[36.21147884 64.59609802 34.85742653 41.27775966 51.19737103]
  [32.21035588 43.95062771 56.87534677 61.61711748 55.18827644]
  [48.04114939 35.02458642 53.51844653 68.80599673 66.27309256]
  [50.57698829 57.38687957 48.69819476 54.42624069 61.62825001]
  [58.17259996 40.45669997 59.93735524 60.26201973 51.56386368]]]
```

3.6 Combining a set of 2-dim. arrays

Make a Python script to combine a set of 2-dimensional arrays using a 3-dimensional array.

Python Code 5: ao2021_s04_05.py

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

# importing Numpy module
import numpy
import numpy.random

# parameters
n_x = 256
n_y = 256
n_z = 100
mean  = 50.0
sigma = 10.0

# creating a data cube from a set of 2-dim. arrays
for i in range (n_z):
    # creating 2-dim. array
    tmp = numpy.random.normal (mean, sigma, (n_x, n_y) )
    # concatenating 2-dim. arrays to make a data cube
    if (i == 0):
        tmp0 = tmp
    elif (i == 1):
        cube = numpy.concatenate ( ([tmp0], [tmp]), axis=0 )
    else:
        cube = numpy.concatenate ( (cube, [tmp]), axis=0 )

# printing information of "cube"
print ("shape of cube =", cube.shape)

# combining 2-dim. arrays using simple average
combined = numpy.mean (cube, axis=0)

# printing information of "combined"
```

```
print ("shape of combined =", combined.shape)

# printing "combined"
print (combined)
```

Execute the script.

```
% chmod a+x ao2021_s04_05.py
% ./ao2021_s04_05.py
shape of cube = (100, 256, 256)
shape of combined = (256, 256)
[[48.39998883 51.02534775 50.93288763 ... 50.20932234 50.34952918
  50.21030055]
 [49.14443851 51.05323776 49.4950031  ... 51.79404407 50.31587349
  48.75288043]
 [51.34230932 49.17000597 51.5521658  ... 48.2451298  49.45015302
  49.45042692]
 ...
 [49.30979485 49.30327984 49.40088584 ... 49.15906046 50.71937278
  50.68412276]
 [50.33011465 50.01663629 50.23422585 ... 50.39055485 50.39483309
  50.81140672]
 [51.26275971 50.69639491 47.94789263 ... 49.86264693 49.34153224
  48.6332567 ]]
```

3.7 Combining a set of 2-dim. arrays using sigma-clipping

Make a Python script to combine a set of 2-dimensional arrays using sigma clipping algorithm.

Python Code 6: ao2021_s04_06.py

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

# importing Numpy module
import numpy
import numpy.random

# importing Astropy module
import astropy.stats

# parameters
n_x = 256
n_y = 256
n_z = 100
mean  = 50.0
sigma = 10.0

# creating a data cube from a set of 2-dim. arrays
for i in range (n_z):
    # creating 2-dim. array
    tmp = numpy.random.normal (mean, sigma, (n_x, n_y) )

    # adding an outlier
    x = int ( numpy.random.uniform (0, n_x) )
    y = int ( numpy.random.uniform (0, n_y) )
    tmp[x][y] += 10000.0

    # concatenating 2-dim. arrays to make a data cube
```

```
    if (i == 0):
        tmp0 = tmp
    elif (i == 1):
        cube = numpy.concatenate ( ([tmp0], [tmp]), axis=0 )
    else:
        cube = numpy.concatenate ( (cube, [tmp]), axis=0 )

# printing information of "cube"
print ("shape of cube =", cube.shape)

# combining 2-dim. arrays using simple average
combined_simple = numpy.mean (cube, axis=0)

# combining 2-dim. arrays using sigma clipping
combined_sigclip, median, stddev \
    = astropy.stats.sigma_clipped_stats (cube, sigma=3.0, maxiters=10, \
                                         cenfunc='mean', stdfunc='std', \
                                         axis=0)

# printing information of "combined"
print ("shape of combined =", combined_sigclip.shape)

# printing "combined"
print (combined_sigclip)

# max and min
print ("min of combined_simple:  %f" % numpy.amin (combined_simple) )
print ("max of combined_simple:  %f" % numpy.amax (combined_simple) )
print ("min of combined_sigclip: %f" % numpy.amin (combined_sigclip) )
print ("max of combined_sigclip: %f" % numpy.amax (combined_sigclip) )
```

Execute the script.

```
% chmod a+x ao2021_s04_06.py
% shape of cube = (100, 256, 256)
shape of combined = (256, 256)
[[51.30496604 49.92742343 50.80874408 ... 49.26393109 50.30000705
  49.25465762]
 [51.00679821 51.08249776 48.32002957 ... 48.93616243 49.29489372
  50.56172346]
 [50.77081668 49.44769144 49.2358886  ... 49.89939948 50.23410372
  49.50036844]
 ...
 [48.53981231 50.28831354 49.7011771  ... 50.81056389 50.89692874
  49.50802914]
 [49.96128127 49.13175586 50.70770407 ... 51.13414082 49.96233375
  49.78380395]
 [50.20332052 49.600536   49.85523152 ... 51.03643621 50.21201602
  51.54813542]]
min of combined_simple:  45.745483
max of combined_simple:  153.250106
min of combined_sigclip: 45.745483
max of combined_sigclip: 54.128528
```

4 Checking data

Make a Python script to check the data for this session.

Python Code 7: ao2021_s04_07.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 = 'TIME-OBS,IMAGETYP,OBJECT,EXPTIME'
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:
        if key in header0:
            value = str (header0[key])
        else:
            value = "__NONE__"
        record += " %8s" % value
```

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

Execute the script and generate a simple observing log.

```
% chmod a+x ao2021_s04_07.py
% ./ao2021_s04_07.py -h
usage: ao2021_s04_07.py [-h] [-k KEYWORD] files [files ...]

Generating a simple observing log

positional arguments:
  files                 FITS files

optional arguments:
  -h, --help            show this help message and exit
  -k KEYWORD, --keyword KEYWORD
                        a list of keyword to check

% ./ao2021_s04_07.py data_ao2021_s04/*.fits
lot_20210210_1020.fits 20:07:18     BIAS     dark       0.0
lot_20210210_1021.fits 20:07:23     BIAS     dark       0.0
lot_20210210_1022.fits 20:07:28     BIAS     dark       0.0
lot_20210210_1023.fits 20:07:33     BIAS     dark       0.0
lot_20210210_1024.fits 20:07:38     BIAS     dark       0.0
lot_20210210_1025.fits 20:07:43     BIAS     dark       0.0
lot_20210210_1026.fits 20:07:48     BIAS     dark       0.0
lot_20210210_1027.fits 20:07:53     BIAS     dark       0.0
lot_20210210_1028.fits 20:07:58     BIAS     dark       0.0
lot_20210210_1029.fits 20:08:03     BIAS     dark       0.0
lot_20210210_1030.fits 20:08:07     DARK     dark      10.0
lot_20210210_1031.fits 20:08:22     DARK     dark      10.0
lot_20210210_1032.fits 20:08:37     DARK     dark      10.0
lot_20210210_1033.fits 20:08:52     DARK     dark      10.0
lot_20210210_1034.fits 20:09:07     DARK     dark      10.0
lot_20210210_1035.fits 20:09:22     DARK     dark      10.0
lot_20210210_1036.fits 20:09:37     DARK     dark      10.0
lot_20210210_1037.fits 20:09:52     DARK     dark      10.0
lot_20210210_1038.fits 20:10:07     DARK     dark      10.0
lot_20210210_1039.fits 20:10:22     DARK     dark      10.0
lot_20210210_1040.fits 20:10:37     BIAS     dark       0.0
lot_20210210_1041.fits 20:10:42     BIAS     dark       0.0
lot_20210210_1042.fits 20:10:47     BIAS     dark       0.0
lot_20210210_1043.fits 20:10:52     BIAS     dark       0.0
lot_20210210_1044.fits 20:10:57     BIAS     dark       0.0
lot_20210210_1045.fits 20:11:02     BIAS     dark       0.0
lot_20210210_1046.fits 20:11:07     BIAS     dark       0.0
lot_20210210_1047.fits 20:11:12     BIAS     dark       0.0
lot_20210210_1048.fits 20:11:17     BIAS     dark       0.0
lot_20210210_1049.fits 20:11:21     BIAS     dark       0.0
lot_20210210_1050.fits 20:11:26     DARK     dark      30.0
lot_20210210_1051.fits 20:12:01     DARK     dark      30.0
lot_20210210_1052.fits 20:12:36     DARK     dark      30.0
lot_20210210_1053.fits 20:13:11     DARK     dark      30.0
lot_20210210_1054.fits 20:13:46     DARK     dark      30.0
lot_20210210_1055.fits 20:14:21     DARK     dark      30.0
lot_20210210_1056.fits 20:14:56     DARK     dark      30.0
```

```
lot_20210210_1057.fits 20:15:33     DARK     dark     30.0
lot_20210210_1058.fits 20:16:08     DARK     dark     30.0
lot_20210210_1059.fits 20:16:45     DARK     dark     30.0
lot_20210210_1060.fits 20:17:20     BIAS     dark      0.0
lot_20210210_1061.fits 20:17:25     BIAS     dark      0.0
lot_20210210_1062.fits 20:17:30     BIAS     dark      0.0
lot_20210210_1063.fits 20:17:35     BIAS     dark      0.0
lot_20210210_1064.fits 20:17:40     BIAS     dark      0.0
lot_20210210_1065.fits 20:17:45     BIAS     dark      0.0
lot_20210210_1066.fits 20:17:50     BIAS     dark      0.0
lot_20210210_1067.fits 20:17:55     BIAS     dark      0.0
lot_20210210_1068.fits 20:18:00     BIAS     dark      0.0
lot_20210210_1069.fits 20:18:05     BIAS     dark      0.0
lot_20210210_1070.fits 20:18:10     DARK     dark     90.0
lot_20210210_1071.fits 20:19:45     DARK     dark     90.0
lot_20210210_1072.fits 20:21:19     DARK     dark     90.0
lot_20210210_1073.fits 20:22:54     DARK     dark     90.0
lot_20210210_1074.fits 20:24:29     DARK     dark     90.0
lot_20210210_1075.fits 20:26:04     DARK     dark     90.0
lot_20210210_1076.fits 20:27:39     DARK     dark     90.0
lot_20210210_1077.fits 20:29:14     DARK     dark     90.0
lot_20210210_1078.fits 20:30:49     DARK     dark     90.0
lot_20210210_1079.fits 20:32:24     DARK     dark     90.0
lot_20210210_1080.fits 20:33:59     BIAS     dark      0.0
lot_20210210_1081.fits 20:34:04     BIAS     dark      0.0
lot_20210210_1082.fits 20:34:09     BIAS     dark      0.0
lot_20210210_1083.fits 20:34:14     BIAS     dark      0.0
lot_20210210_1084.fits 20:34:19     BIAS     dark      0.0
lot_20210210_1085.fits 20:34:24     BIAS     dark      0.0
lot_20210210_1086.fits 20:34:29     BIAS     dark      0.0
lot_20210210_1087.fits 20:34:34     BIAS     dark      0.0
lot_20210210_1088.fits 20:34:39     BIAS     dark      0.0
lot_20210210_1089.fits 20:34:44     BIAS     dark      0.0
lot_20210210_1090.fits 20:34:49     DARK     dark    300.0
lot_20210210_1091.fits 20:39:54     DARK     dark    300.0
lot_20210210_1092.fits 20:44:59     DARK     dark    300.0
lot_20210210_1093.fits 20:50:04     DARK     dark    300.0
lot_20210210_1094.fits 20:55:09     DARK     dark    300.0
lot_20210210_1095.fits 21:00:14     DARK     dark    300.0
lot_20210210_1096.fits 21:05:19     DARK     dark    300.0
lot_20210210_1097.fits 21:10:24     DARK     dark    300.0
lot_20210210_1098.fits 21:15:29     DARK     dark    300.0
lot_20210210_1099.fits 21:20:34     DARK     dark    300.0
lot_20210210_1100.fits 21:25:39     BIAS     dark      0.0
lot_20210210_1101.fits 21:25:44     BIAS     dark      0.0
lot_20210210_1102.fits 21:25:49     BIAS     dark      0.0
lot_20210210_1103.fits 21:25:54     BIAS     dark      0.0
lot_20210210_1104.fits 21:25:59     BIAS     dark      0.0
lot_20210210_1105.fits 21:26:04     BIAS     dark      0.0
lot_20210210_1106.fits 21:26:09     BIAS     dark      0.0
lot_20210210_1107.fits 21:26:14     BIAS     dark      0.0
lot_20210210_1108.fits 21:26:19     BIAS     dark      0.0
lot_20210210_1109.fits 21:26:24     BIAS     dark      0.0
lot_20210210_1110.fits 21:26:29     DARK     dark    900.0
lot_20210210_1111.fits 21:41:34     DARK     dark    900.0
lot_20210210_1112.fits 21:56:39     DARK     dark    900.0
lot_20210210_1113.fits 22:11:43     DARK     dark    900.0
lot_20210210_1114.fits 22:26:48     DARK     dark    900.0
lot_20210210_1115.fits 22:41:53     DARK     dark    900.0
```

```
lot_20210210_1116.fits 22:56:58     DARK     dark    900.0
lot_20210210_1117.fits 23:12:03     DARK     dark    900.0
lot_20210210_1118.fits 23:27:08     DARK     dark    900.0
lot_20210210_1119.fits 23:42:13     DARK     dark    900.0
lot_20210210_1120.fits 23:57:18     BIAS     dark      0.0
lot_20210210_1121.fits 23:57:23     BIAS     dark      0.0
lot_20210210_1122.fits 23:57:28     BIAS     dark      0.0
lot_20210210_1123.fits 23:57:33     BIAS     dark      0.0
lot_20210210_1124.fits 23:57:38     BIAS     dark      0.0
lot_20210210_1125.fits 23:57:43     BIAS     dark      0.0
lot_20210210_1126.fits 23:57:48     BIAS     dark      0.0
lot_20210210_1127.fits 23:57:53     BIAS     dark      0.0
lot_20210210_1128.fits 23:57:58     BIAS     dark      0.0
lot_20210210_1129.fits 23:58:03     BIAS     dark      0.0
lot_20210210_1130.fits 23:58:08     DARK     dark   3600.0
lot_20210210_1131.fits 00:58:13     DARK     dark   3600.0
lot_20210210_1132.fits 01:58:25     DARK     dark   3600.0
lot_20210210_1133.fits 02:58:30     DARK     dark   3600.0
lot_20210210_1134.fits 03:58:42     DARK     dark   3600.0
lot_20210210_1135.fits 04:58:47     DARK     dark   3600.0
lot_20210210_1136.fits 05:58:52     DARK     dark   3600.0
lot_20210210_1137.fits 06:58:57     DARK     dark   3600.0
lot_20210210_1138.fits 07:59:03     DARK     dark   3600.0
lot_20210210_1139.fits 08:59:08     DARK     dark   3600.0
lot_20210210_1140.fits 09:59:13     BIAS     dark      0.0
lot_20210210_1141.fits 09:59:21     BIAS     dark      0.0
lot_20210210_1142.fits 09:59:26     BIAS     dark      0.0
lot_20210210_1143.fits 09:59:31     BIAS     dark      0.0
lot_20210210_1144.fits 09:59:36     BIAS     dark      0.0
lot_20210210_1145.fits 09:59:41     BIAS     dark      0.0
lot_20210210_1146.fits 09:59:46     BIAS     dark      0.0
lot_20210210_1147.fits 09:59:51     BIAS     dark      0.0
lot_20210210_1148.fits 09:59:56     BIAS     dark      0.0
lot_20210210_1149.fits 10:00:01     BIAS     dark      0.0
```

The data for this session are bias and dark frames.

5 Calculating statistical values of dark frames

Make a Python script to calculate statistical values of dark frames.

Python Code 8: ao2021_s04_08.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']
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

# printing header
print ("%s" % '-' * 78)
print ("%-24s %8s %8s %8s %8s %8s %8s" \
       % ("file name", "n_pix", "mean", "median", "stddev", "min", "max") )
print ("%s" % '=' * 78)

# 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

    # image data of primary HDU
    data0 = hdu0.data

    # closing FITS file
    hdu_list.close ()

    # flattening data
    data_1d = data0.flatten ()

    # if rejection algorithm is used, then do rejection check
    if (rejection == 'sigclip'):
        clipped, lower, upper \
            = scipy.stats.sigmaclip (data_1d, low=threshold, high=threshold)
    elif (rejection == 'none'):
        clipped = data_1d
```

```
    # 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 ("%-24s %8d %8.2f %8.2f %8.2f %8.2f %8.2f" \
           % (filename, n_pix, mean, median, stddev, vmin, vmax) )

# printing footer
print ("%s" % '=' * 78)
```

Calculate statistical values of 3600-sec dark frames.

```
% chmod a+x ao2021_s04_08.py
% ./ao2021_s04_08.py data_ao2021_s04/lot_20210210_113?.fits
------------------------------------------------------------------------------
file name                    n_pix     mean   median   stddev      min      max
==============================================================================
lot_20210210_1130.fits     4194304   608.88   608.00    44.39   569.00 42812.00
lot_20210210_1131.fits     4194304   608.55   608.00    60.20   568.00 65535.00
lot_20210210_1132.fits     4194304   608.31   608.00    78.25   567.00 65535.00
lot_20210210_1133.fits     4194304   608.98   608.00    47.45   568.00 43170.00
lot_20210210_1134.fits     4194304   608.35   608.00    44.77   566.00 42998.00
lot_20210210_1135.fits     4194304   608.62   608.00    46.25   569.00 42902.00
lot_20210210_1136.fits     4194304   607.99   607.00    58.59   565.00 65535.00
lot_20210210_1137.fits     4194304   608.61   608.00    53.92   567.00 42649.00
lot_20210210_1138.fits     4194304   608.93   608.00    50.95   569.00 42733.00
lot_20210210_1139.fits     4194304   608.34   608.00    63.10   565.00 65535.00
==============================================================================
```

3600-sec dark frames seem to have comic ray hits. Try to use sigma clipping.

```
% ./ao2021_s04_08.py -h
usage: ao2021_s04_08.py [-h] [-r {none,sigclip}] [-t THRESHOLD]
                        files [files ...]

Calculating statistical values

positional arguments:
  files                 FITS files

optional arguments:
  -h, --help            show this help message and exit
  -r {none,sigclip}, --rejection {none,sigclip}
                        outlier rejection algorithm
  -t THRESHOLD, --threshold THRESHOLD
                        rejection threshold in sigma

% ./ao2021_s04_08.py -r sigclip data_ao2021_s04/lot_20210210_113?.fits
------------------------------------------------------------------------------
file name                    n_pix     mean   median   stddev      min      max
==============================================================================
```

```
lot_20210210_1130.fits    4185911   608.17   608.00     8.08   576.00   640.00
lot_20210210_1131.fits    4185670   607.75   608.00     8.07   576.00   640.00
lot_20210210_1132.fits    4185652   607.52   607.00     8.06   576.00   639.00
lot_20210210_1133.fits    4185291   608.19   608.00     8.06   576.00   640.00
lot_20210210_1134.fits    4185836   607.67   608.00     8.06   576.00   639.00
lot_20210210_1135.fits    4185854   607.92   608.00     8.06   576.00   640.00
lot_20210210_1136.fits    4185289   607.21   607.00     8.07   575.00   639.00
lot_20210210_1137.fits    4185682   607.85   608.00     8.06   576.00   640.00
lot_20210210_1138.fits    4185600   608.17   608.00     8.06   576.00   640.00
lot_20210210_1139.fits    4185650   607.61   608.00     8.07   576.00   639.00
================================================================================
```

6 Visualising a dark frame

Make a Python script to visualise a dark frame.

Python Code 9: ao2021_s04_09.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)

# adding arguments
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

# 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='hot')
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_s04_09.py
% ./ao2021_s04_09.py -i data_ao2021_s04/lot_20210210_1130.fits -o 1130.png
data_ao2021_s04/lot_20210210_1130.fits ==> 1130.png
% ls -l 1130.png
-rw-r--r--  1 daisuke  taiwan  187304 Mar  9 20:18 1130.png
```

Show the image. (Fig. 3) Because of some pixels with large pixel values, we cannot see the detailed structure of the image.

```
% feh -dF 1130.png
```

Modify the script to adjust the range of pixel values to display.

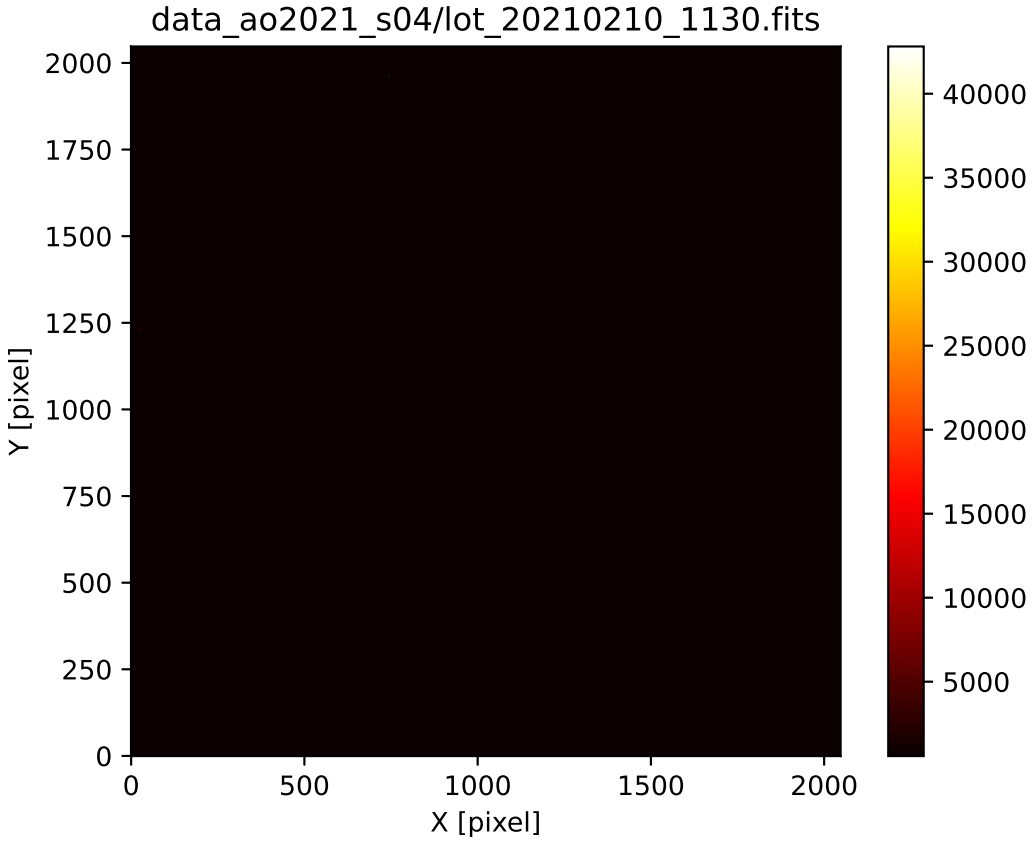

Figure 3: The image of the file `lot_20210210_1130.fits`.

Python Code 10: ao2021_s04_10.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)

# 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 ('-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

# 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='hot', 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_s04_10.py
% ./ao2021_s04_10.py -i data_ao2021_s04/lot_20210210_1130.fits -o 1130_2.png \
? -a 550 -b 650
data_ao2021_s04/lot_20210210_1130.fits ==> 1130_2.png
% ls -l 1130_2.png
-rw-r--r--  1 daisuke  taiwan  3920178 Mar  9 20:26 1130_2.png
```

Show the image. (Fig. 4) Now, it looks better.

```
% feh -dF 1130_2.png
```

Add a function to choose a colour map.

Python Code 11: ao2021_s04_11.py

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

# importing argparse module
import argparse

# importing sys module
import sys

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

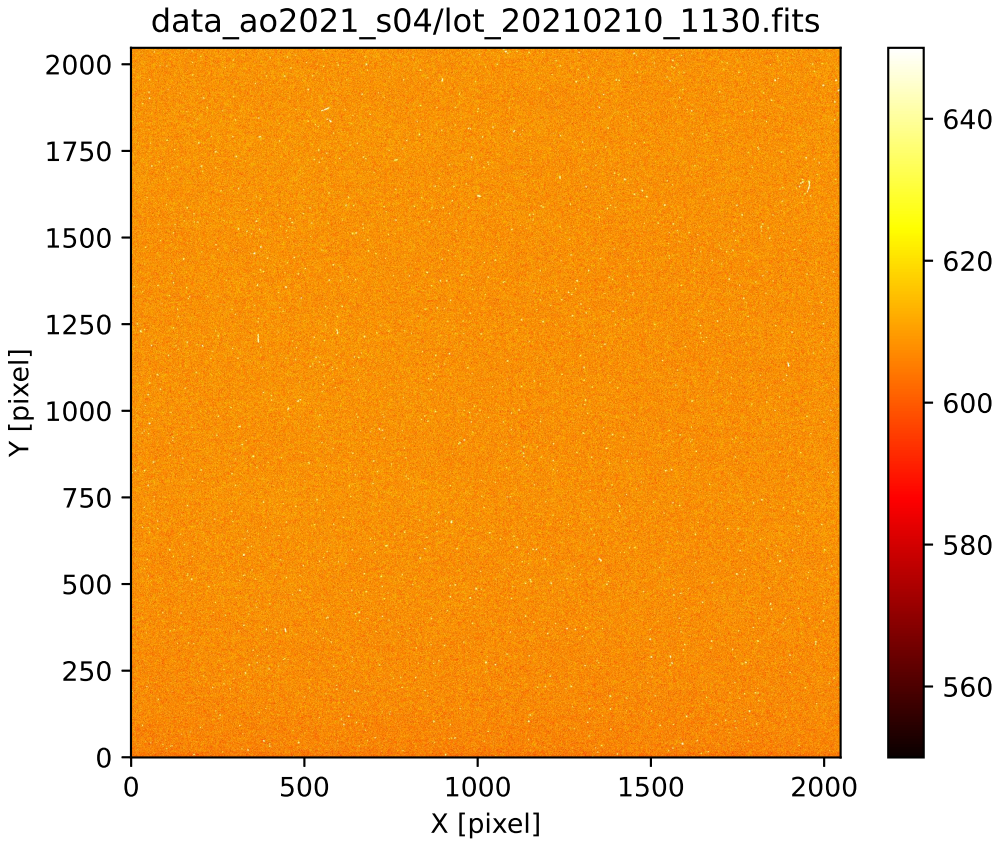

Figure 4: The image of the file `lot_20210210_1130.fits` using the range [550, 650].

```
# 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)
```

Use a colour map "viridis" to generate a PNG file.

```
% chmod a+x ao2021_s04_11.py
% ./ao2021_s04_11.py -h
usage: ao2021_s04_11.py [-h] [-a MIN] [-b MAX]
                        [-c {viridis,plasma,inferno,magma,cividis,gray,bone,pink
,cool,hot,spring,summer,autumn,winter}]
                        [-i INPUT] [-o OUTPUT]

Reading a FITS file and making a PNG file

optional arguments:
  -h, --help            show this help message and exit
  -a MIN, --min MIN     minimum pixel value
  -b MAX, --max MAX     maximum pixel value
  -c {viridis,plasma,inferno,magma,cividis,gray,bone,pink,cool,hot,spring,summer
,autumn,winter}, --cmap {viridis,plasma,inferno,magma,cividis,gray,bone,pink,coo
l,hot,spring,summer,autumn,winter}
                        maximum pixel value
  -i INPUT, --input INPUT
                        intput FITS file
  -o OUTPUT, --output OUTPUT
                        output image file

% ./ao2021_s04_11.py -i data_ao2021_s04/lot_20210210_1130.fits -o 1130_3.png \
? -a 550 -b 650 -c viridis
data_ao2021_s04/lot_20210210_1130.fits ==> 1130_3.png
% ls -l 1130_3.png
-rw-r--r--  1 daisuke  taiwan  6391390 Mar  9 20:43 1130_3.png
```

Display the image. (Fig. 5)

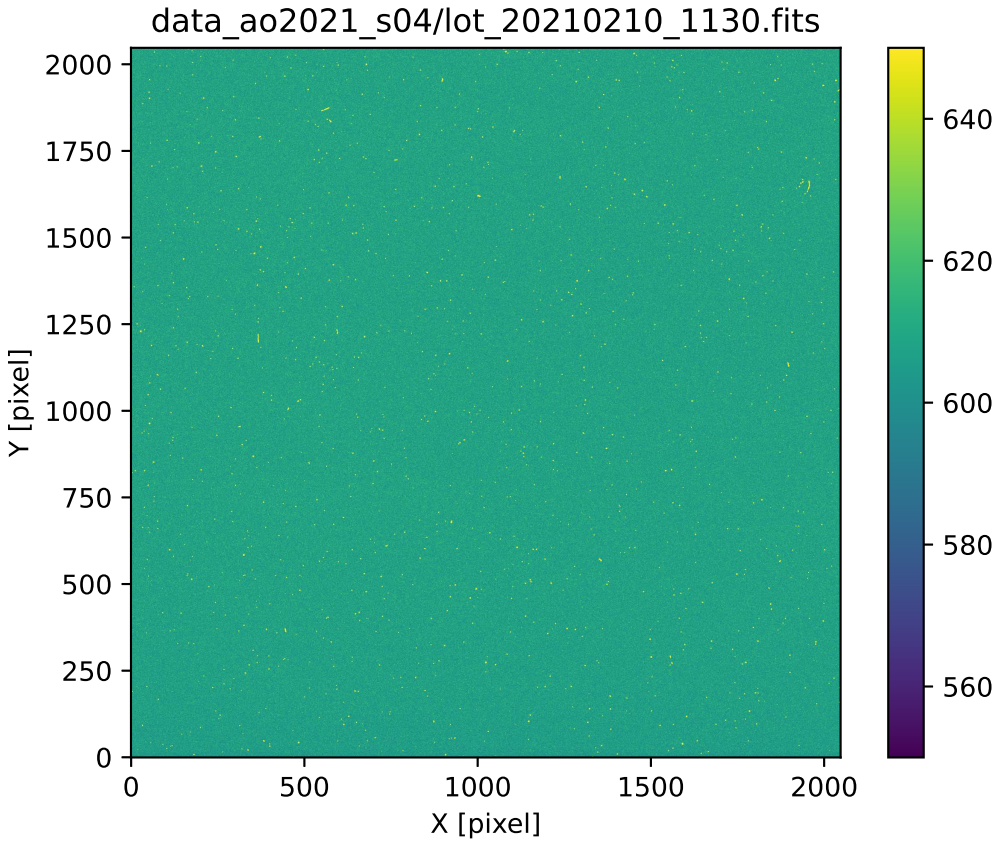

Figure 5: The image of the file `lot_20210210_1130.fits` using the range [550, 650] with colour map "viridis".

7 Combining dark frames

In order to do the CCD data reduction, we need to carry out dark subtraction for object frames. For dark subtraction, multiple dark frames of the same exposure time must be combined to reduce the noise. Make a Python script to combine dark frames. Here is an example.

Python Code 12: ao2021_s04_12.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']
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 ('-n', '--maxiters', type=int, default=10, \
                     help='maximum number of iterations')
parser.add_argument ('-c', '--cenfunc', choices=list_cenfunc, default='mean', \
                     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

# 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

# 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 only for the first FITS file
    if (i == 0):
        header0 = hdu0.header

    # 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
```

```
# 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
header0['history'] = "FITS file created by the command \"%s\"" % (command)
header0['history'] = "Updated on %s" % (now)
header0['comment'] = "List of combined files:"
for fits in file_input:
    header0['comment'] = "  %s" % (fits)
header0['comment'] = "Options given:"
header0['comment'] = "  rejection = %s" % (rejection)
header0['comment'] = "  threshold = %f sigma" % (threshold)
header0['comment'] = "  maxiters  = %d" % (maxiters)
header0['comment'] = "  cenfunc   = %s" % (cenfunc)

# writing a new FITS file
astropy.io.fits.writeto (file_output, combined, header=header0)
```

Run the script to combine 3600-sec dark frames without using sigma clipping.

```
% chmod a+x ao2021_s04_12.py
% ./ao2021_s04_12.py -h
usage: ao2021_s04_12.py [-h] [-r {none,sigclip}] [-t THRESHOLD] [-n MAXITERS]
                        [-o OUTPUT]
                        files [files ...]

Combining images

positional arguments:
  files                 input FITS files

optional arguments:
  -h, --help            show this help message and exit
  -r {none,sigclip}, --rejection {none,sigclip}
                        outlier rejection algorithm
  -t THRESHOLD, --threshold THRESHOLD
                        rejection threshold in sigma
  -n MAXITERS, --maxiters MAXITERS
                        maximum number of iterations
  -o OUTPUT, --output OUTPUT
                        output FITS file

% ./ao2021_s04_12.py -o dark3600_simple.fits \
? data_ao2021_s04/lot_20210210_113?.fits
% ls -l dark3600_simple.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar  9 22:25 dark3600_simple.fits
```

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

Python Code 13: ao2021_s04_13.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 a FITS file.

```
% chmod a+x ao2021_s04_13.py
% ./ao2021_s04_13.py dark3600_simple.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_s04_12.py"
HISTORY Updated on 2021-03-09T22:25:36.047962
COMMENT List of combined files:
COMMENT   data_ao2021_s04/lot_20210210_1130.fits
COMMENT   data_ao2021_s04/lot_20210210_1131.fits
COMMENT   data_ao2021_s04/lot_20210210_1132.fits
COMMENT   data_ao2021_s04/lot_20210210_1133.fits
COMMENT   data_ao2021_s04/lot_20210210_1134.fits
COMMENT   data_ao2021_s04/lot_20210210_1135.fits
COMMENT   data_ao2021_s04/lot_20210210_1136.fits
COMMENT   data_ao2021_s04/lot_20210210_1137.fits
COMMENT   data_ao2021_s04/lot_20210210_1138.fits
COMMENT   data_ao2021_s04/lot_20210210_1139.fits
COMMENT Options given:
COMMENT   rejection = none
COMMENT   threshold = 4.000000 sigma
COMMENT   maxiters  = 10
COMMENT   cenfunc   = mean
```

We can confirm that new comments are included in the header of a new FITS file.
Next, combine 3600-sec dark frames using sigma clipping.

```
% ./ao2021_s04_12.py -o dark3600_sigclip.fits -r sigclip -t 3 -c median \
? data_ao2021_s04/lot_20210210_113?.fits
% ls -l dark3600_sigclip.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar  9 22:26 dark3600_sigclip.fits
```

Print the header of the file.

```
% ./ao2021_s04_13.py dark3600_sigclip.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_s04_12.py"
HISTORY Updated on 2021-03-09T22:26:50.411615
COMMENT List of combined files:
COMMENT   data_ao2021_s04/lot_20210210_1130.fits
COMMENT   data_ao2021_s04/lot_20210210_1131.fits
COMMENT   data_ao2021_s04/lot_20210210_1132.fits
COMMENT   data_ao2021_s04/lot_20210210_1133.fits
COMMENT   data_ao2021_s04/lot_20210210_1134.fits
COMMENT   data_ao2021_s04/lot_20210210_1135.fits
COMMENT   data_ao2021_s04/lot_20210210_1136.fits
COMMENT   data_ao2021_s04/lot_20210210_1137.fits
COMMENT   data_ao2021_s04/lot_20210210_1138.fits
COMMENT   data_ao2021_s04/lot_20210210_1139.fits
COMMENT Options given:
COMMENT   rejection = sigclip
```

```
COMMENT   threshold = 3.000000 sigma
COMMENT   maxiters  = 10
COMMENT   cenfunc   = median
```

Compare these two files.

```
% ./ao2021_s04_08.py dark3600_*.fits
--------------------------------------------------------------------------------
file name                    n_pix     mean   median   stddev      min      max
================================================================================
dark3600_sigclip.fits      4194304   607.87   607.80    25.86   592.50 42845.40
dark3600_simple.fits       4194304   608.55   607.90    30.10   592.50 42845.40
================================================================================
```

Show the image of `dark3600_simple.fits` (Fig. 6) and `dark3600_sigclip.fits` (Fig. 7). Many cosmic rays are removed by sigma clipping.

```
% ./ao2021_s04_11.py -a 590 -b 640 -c viridis -i dark3600_simple.fits \
? -o dark3600_simple.png
dark3600_simple.fits ==> dark3600_simple.png
% ls -l dark3600_simple.png
-rw-r--r--  1 daisuke  taiwan  4292724 Mar  9 22:44 dark3600_simple.png
% ./ao2021_s04_11.py -a 590 -b 640 -c viridis -i dark3600_sigclip.fits \
? -o dark3600_sigclip.png
dark3600_sigclip.fits ==> dark3600_sigclip.png
% ls -l dark3600_sigclip.png
-rw-r--r--  1 daisuke  taiwan  4055731 Mar  9 22:47 dark3600_sigclip.png
% feh -dF dark3600_*.png
```

8 Estimating dark current generation rate

8.1 Combining dark frames

Combine dark frames.

```
% ./ao2021_s04_12.py -r sigclip -t 3 -c median -o dark0010.fits */*_103?.fits
% ./ao2021_s04_12.py -r sigclip -t 3 -c median -o dark0030.fits */*_105?.fits
% ./ao2021_s04_12.py -r sigclip -t 3 -c median -o dark0090.fits */*_107?.fits
% ./ao2021_s04_12.py -r sigclip -t 3 -c median -o dark0300.fits */*_109?.fits
% ./ao2021_s04_12.py -r sigclip -t 3 -c median -o dark0900.fits */*_111?.fits
% ./ao2021_s04_12.py -r sigclip -t 3 -c median -o dark3600.fits */*_113?.fits
% ls -l dark????.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar  9 23:01 dark0010.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar  9 23:02 dark0030.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar  9 23:02 dark0090.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar  9 23:02 dark0300.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar  9 23:03 dark0900.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar  9 23:03 dark3600.fits
```

8.2 Calculating mean values of combined dark frames

Calculate mean values of combined dark frames.

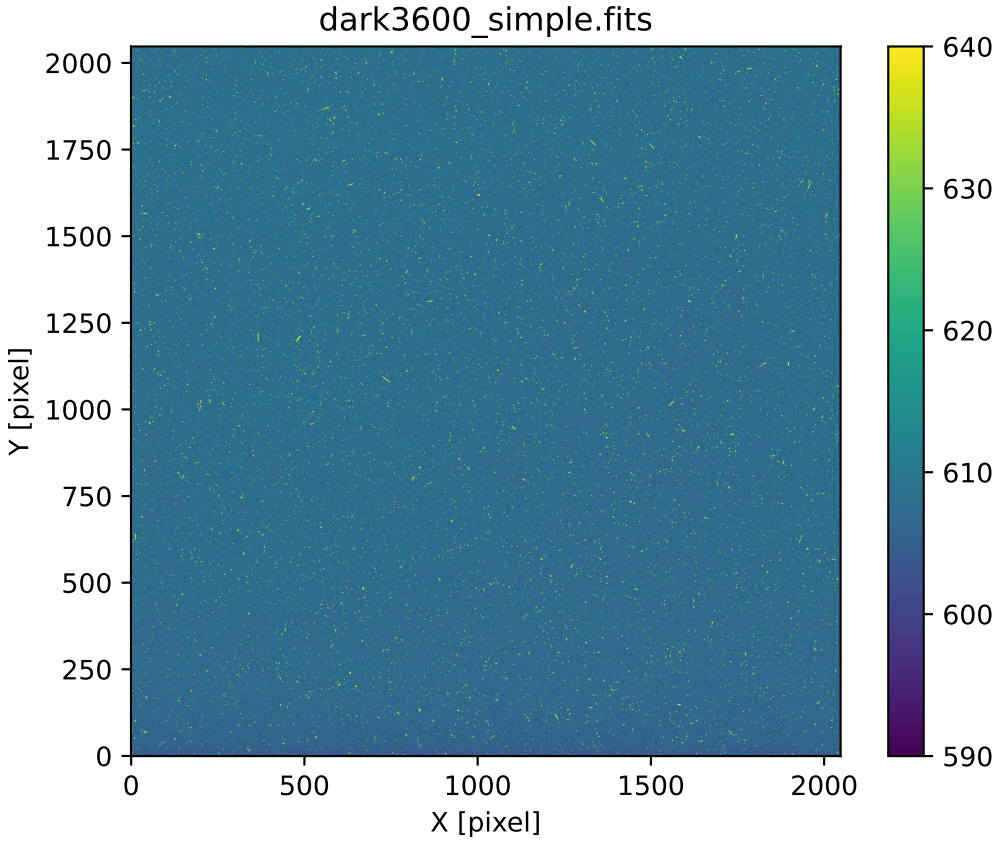

Figure 6: Combined 3600-sec dark frame using simple mean.

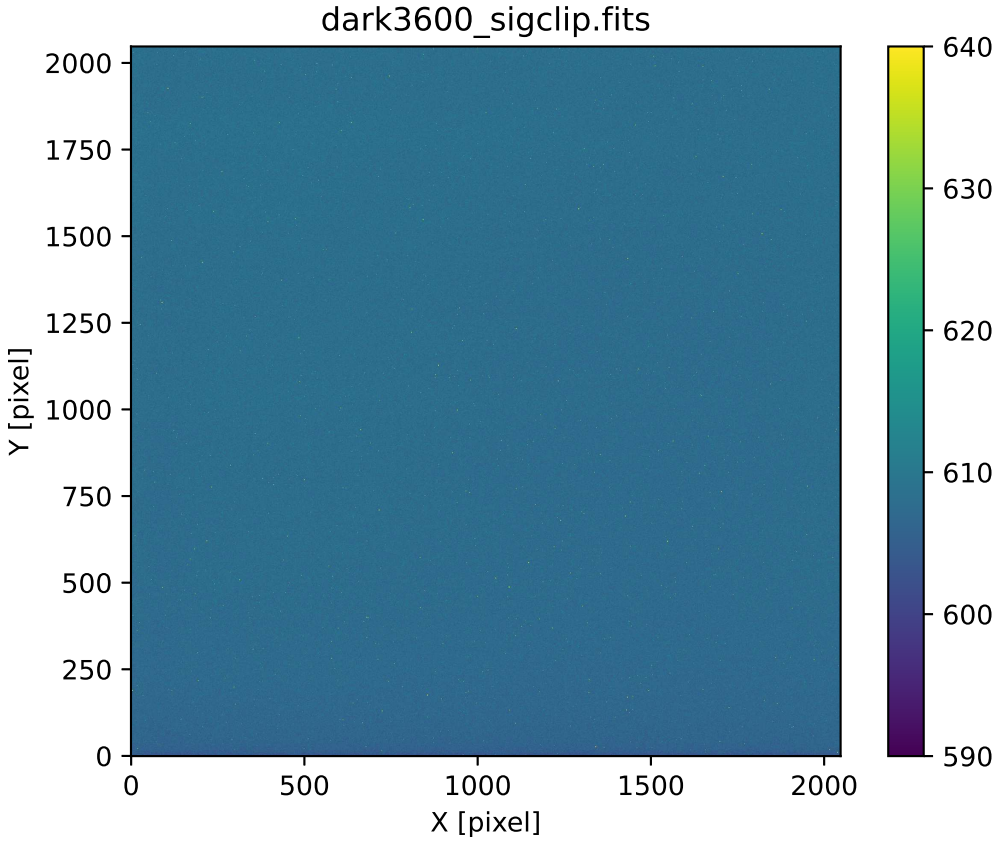

Figure 7: Combined 3600-sec dark frame using sigma clipping.

```
% ./ao2021_s04_08.py -r sigclip dark????.fits
--------------------------------------------------------------------------------
file name                    n_pix      mean    median    stddev       min       max
================================================================================
dark0010.fits              4193636    606.44    606.50      2.60    596.10    616.80
dark0030.fits              4193583    606.54    606.50      2.58    596.22    616.80
dark0090.fits              4193563    606.53    606.50      2.60    596.20    616.90
dark0300.fits              4193456    606.42    606.40      2.60    596.10    616.80
dark0900.fits              4192607    606.62    606.60      2.59    596.30    616.90
dark3600.fits              4190196    607.81    607.80      2.66    597.20    618.44
================================================================================
```

8.3 Making a plot

Make a plot of exposure time vs. mean pixel value of dark frame.

Python Code 14: ao2021_s04_14.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 matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# making empty numpy arrays for plotting
data_exptime = numpy.array ([], dtype='float64')
data_mean    = numpy.array ([], dtype='float64')
data_stddev  = numpy.array ([], dtype='float64')

# construction of parser object
desc = 'Making a plot of exposure time vs mean pixel value of dark frame'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_rejection = ['none', 'sigclip']
list_cenfunc   = ['mean', 'median']
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 ('-n', '--maxiters', type=int, default=10, \
                     help='maximum number of iterations')
parser.add_argument ('-c', '--cenfunc', choices=list_cenfunc, default='mean', \
                     help='method to estimate centre value')
parser.add_argument ('-o', '--output', default='dark.png', \
```

```
                    help='output image 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

# 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 PNG or PDF or PS file, then stop the script
if not ( (file_output[-4:] == '.png') or (file_output[-4:] == '.pdf') \
         or (file_output[-3:] == '.ps') ):
    # printing error message
    print ("Output file must be PNG or PDF or PS file!")
    # exit the script
    sys.exit ()

# 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

    # reading data
    data0 = hdu0.data

    # closing FITS file
    hdu_list.close ()

    # exposure time
    exptime = header0['EXPTIME']

    # mean value
    if (rejection == 'sigclip'):
        mean, median, stddev \
            = astropy.stats.sigma_clipped_stats (data0, sigma=threshold, \
                                                 maxiters=maxiters, \
                                                 cenfunc=cenfunc, \
```

```
                                                    stdfunc='std')
    elif (rejection == 'none'):
        mean   = numpy.nanmean (data0)
        stddev = numpy.nanstd (data0)

    # appending data to numpy arrays
    data_exptime = numpy.append (data_exptime, exptime)
    data_mean    = numpy.append (data_mean, mean)
    data_stddev  = numpy.append (data_stddev, stddev)

# printing data which will be used for plotting
print ("data_exptime:")
print (data_exptime)
print ("data_mean:")
print (data_mean)
print ("data_stddev:")
print (data_stddev)

# plotting using Matplotlib

# 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 ('Exposure Time [sec]')
ax.set_ylabel ('Mean Pixel Value [ADU]')

# plotting a figure
ax.set_title ('Dark Current Generation Rate')
ax.errorbar (data_exptime, data_mean, yerr=data_stddev, \
             fmt='bo', ecolor='black', capsize=5, label='Dark Current')
ax.legend ()

# saving the figure to a file
fig.savefig (file_output, dpi=450)
```

Run the script, and generate a plot.

```
% ./ao2021_s04_14.py -h
usage: ao2021_s04_14.py [-h] [-r {none,sigclip}] [-t THRESHOLD] [-n MAXITERS]
                        [-c {mean,median}] [-o OUTPUT]
                        files [files ...]

Making a plot of exposure time vs mean pixel value of dark frame

positional arguments:
  files                 input FITS files

optional arguments:
  -h, --help            show this help message and exit
  -r {none,sigclip}, --rejection {none,sigclip}
                        outlier rejection algorithm
  -t THRESHOLD, --threshold THRESHOLD
                        rejection threshold in sigma
  -n MAXITERS, --maxiters MAXITERS
```

```
                        maximum number of iterations
  -c {mean,median}, --cenfunc {mean,median}
                        method to estimate centre value
  -o OUTPUT, --output OUTPUT
                        output image file

% ./ao2021_s04_14.py -r sigclip -o darkrate.png dark????.fits
data_exptime:
[  10.   30.   90.  300.  900. 3600.]
data_mean:
[606.44246003 606.53848506 606.52723527 606.41802709 606.62461922
 607.81367505]
data_stddev:
[2.5955225  2.58302423 2.59709639 2.60099502 2.58791261 2.65817185]
% ls -l darkrate.png
-rw-r--r--  1 daisuke  taiwan  143562 Mar  9 23:41 darkrate.png
```

Show the plot. (Fig. 8)

```
% feh -dF darkrate.png
```

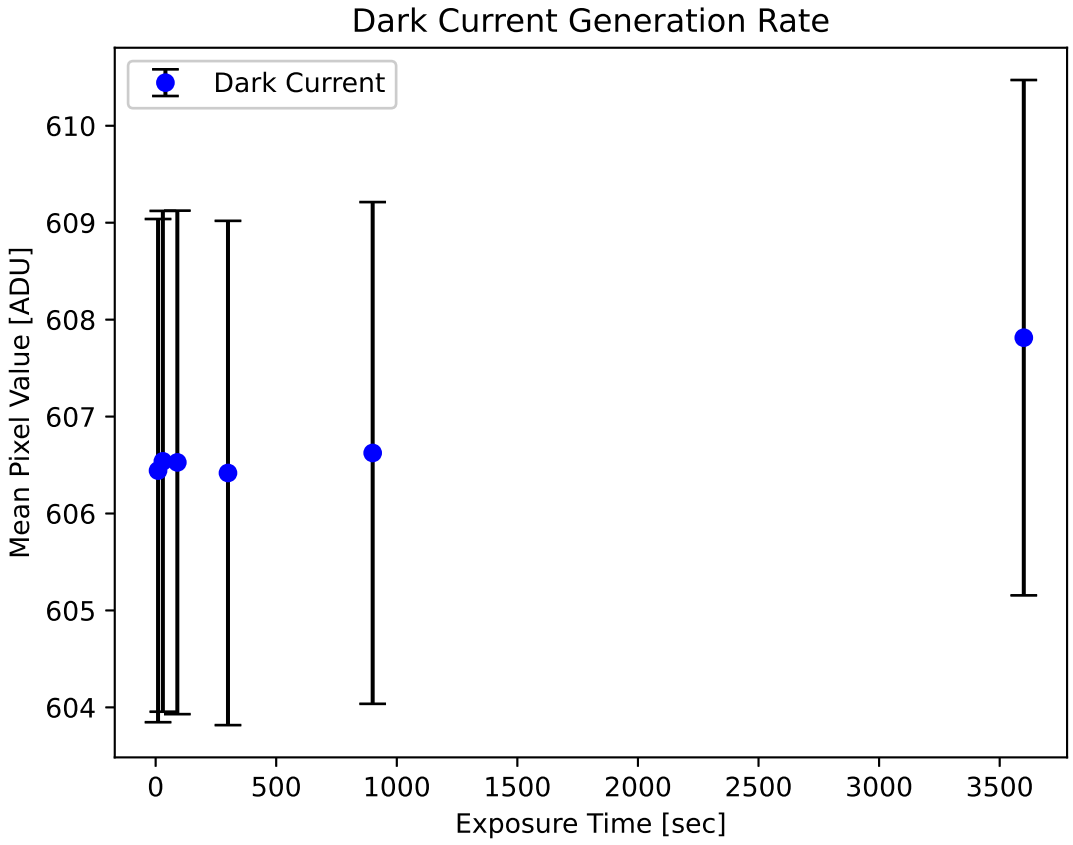

Figure 8: The plot of exposure time vs. mean pixel value of dark frame.

8.4 Fitting using least squares method

Modify the previous script, and carry out fitting using least squares method.

Python Code 15: ao2021_s04_15.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing numpy module
import numpy

# importing scipy module
import scipy.optimize

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

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

# making empty numpy arrays for plotting
data_exptime = numpy.array ([], dtype='float64')
data_mean    = numpy.array ([], dtype='float64')
data_stddev  = numpy.array ([], dtype='float64')

# construction of parser object
desc = 'Making a plot of exposure time vs mean pixel value of dark frame'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_rejection = ['none', 'sigclip']
list_cenfunc   = ['mean', 'median']
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 ('-n', '--maxiters', type=int, default=10, \
                     help='maximum number of iterations')
parser.add_argument ('-c', '--cenfunc', choices=list_cenfunc, default='mean', \
                     help='method to estimate centre value')
parser.add_argument ('-o', '--output', default='dark.png', \
                     help='output image 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

# 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 PNG or PDF or PS file, then stop the script
if not ( (file_output[-4:] == '.png') or (file_output[-4:] == '.pdf') \
         or (file_output[-3:] == '.ps') ):
    # printing error message
    print ("Output file must be PNG or PDF or PS file!")
    # exit the script
    sys.exit ()

# 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

    # reading data
    data0 = hdu0.data

    # closing FITS file
    hdu_list.close ()

    # exposure time
    exptime = header0['EXPTIME']

    # mean value
    if (rejection == 'sigclip'):
        mean, median, stddev \
            = astropy.stats.sigma_clipped_stats (data0, sigma=threshold, \
                                                 maxiters=maxiters, \
                                                 cenfunc=cenfunc, \
                                                 stdfunc='std')
    elif (rejection == 'none'):
        mean   = numpy.nanmean (data0)
        stddev = numpy.nanstd (data0)

    # appending data to numpy arrays
    data_exptime = numpy.append (data_exptime, exptime)
    data_mean    = numpy.append (data_mean, mean)
    data_stddev  = numpy.append (data_stddev, stddev)

# printing data which will be used for plotting
```

```
print ("data_exptime:")
print (data_exptime)
print ("data_mean:")
print (data_mean)
print ("data_stddev:")
print (data_stddev)

# least-squares method

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

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

# least-squares fitting
popt, pcov = scipy.optimize.curve_fit (func, data_exptime, data_mean, \
                                       p0=(a,b), sigma=data_stddev)

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

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

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

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

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

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

# fitted line
fitted_x = numpy.linspace (0.0, 3600.0, 10**6)
fitted_y = a_fitted * fitted_x + b_fitted

# plotting using Matplotlib

# 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 ('Exposure Time [sec]')
ax.set_ylabel ('Mean Pixel Value [ADU]')

# plotting a figure
ax.set_title ('Dark Current Generation Rate')
ax.errorbar (data_exptime, data_mean, yerr=data_stddev, \
             fmt='bo', ecolor='black', capsize=5, label='Dark current')
ax.plot (fitted_x, fitted_y, 'r--', label='Least-squares fitting by scipy')
ax.legend ()

# saving the figure to a file
fig.savefig (file_output, dpi=450)
```

Execute the script, and carry out least squares fitting.

```
% ./ao2021_s04_15.py -h
usage: ao2021_s04_15.py [-h] [-r {none,sigclip}] [-t THRESHOLD] [-n MAXITERS]
                        [-c {mean,median}] [-o OUTPUT]
                        files [files ...]

Making a plot of exposure time vs mean pixel value of dark frame

positional arguments:
  files                 input FITS files

optional arguments:
  -h, --help            show this help message and exit
  -r {none,sigclip}, --rejection {none,sigclip}
                        outlier rejection algorithm
  -t THRESHOLD, --threshold THRESHOLD
                        rejection threshold in sigma
  -n MAXITERS, --maxiters MAXITERS
                        maximum number of iterations
  -c {mean,median}, --cenfunc {mean,median}
                        method to estimate centre value
  -o OUTPUT, --output OUTPUT
                        output image file

% ./ao2021_s04_15.py -r sigclip -o darkrate_fitting.png dark????.fits
data_exptime:
[  10.   30.   90.  300.  900. 3600.]
data_mean:
[606.44246003 606.53848506 606.52723527 606.41802709 606.62461922
 607.81367505]
data_stddev:
[2.5955225  2.58302423 2.59709639 2.60099502 2.58791261 2.65817185]
popt:
[3.76052360e-04 6.06418214e+02]
pcov:
[[ 1.32049023e-09 -1.05543204e-06]
 [-1.05543204e-06  2.93827320e-03]]
dof = 4
reduced chi^2 = 0.01247278418761193
a = 0.000376 +/- 0.000036 (9.663162 %)
b = 606.418214 +/- 0.054206 (0.008939 %)
```

```
% ls -l darkrate_fitting.png
-rw-r--r--  1 daisuke  taiwan  176099 Mar 10 00:01 darkrate_fitting.png
```

The dark current generation rate is estimated to be $3.76 \pm 0.36 \times 10^{-4}$ ADU/sec/pixel.
Show the plot. (Fig. 9)

```
% feh -dF darkrate_fitting.png
```

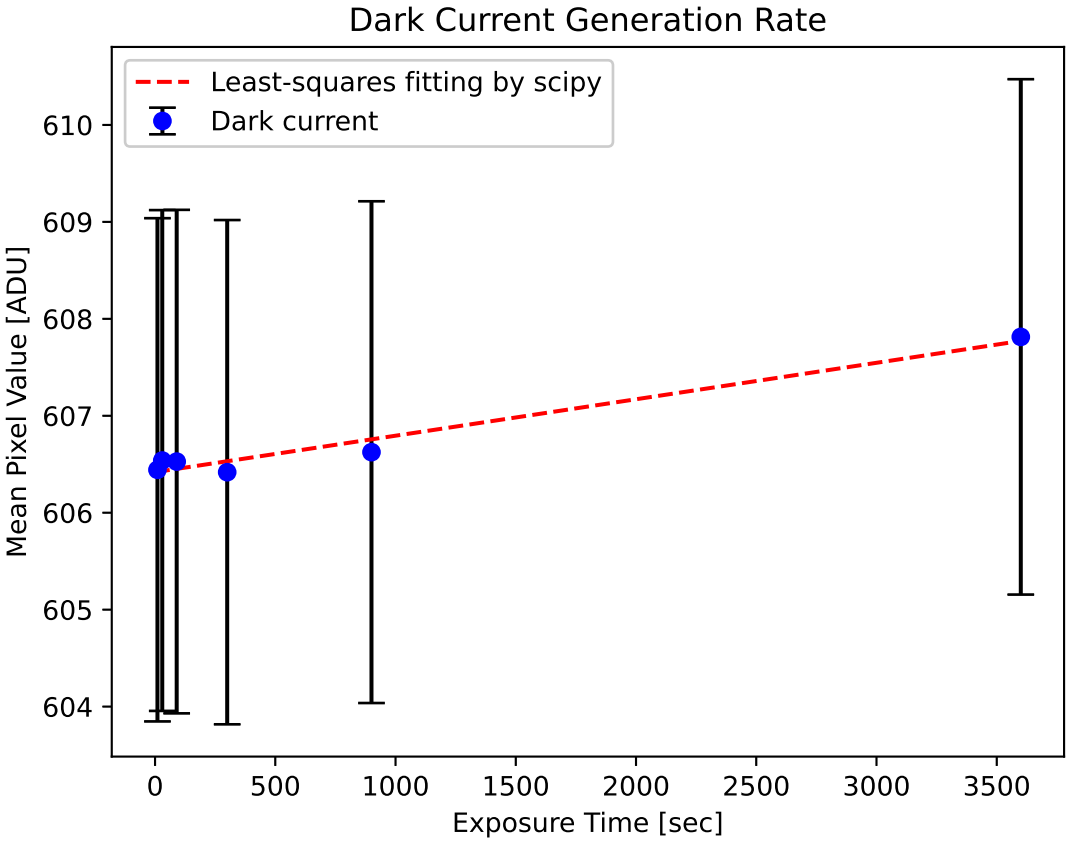

Figure 9: The result of least squares fitting.

9 For your training

- Read chapter 4 of "Handbook of CCD Astronomy" and learn about dark 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 Numpy arrays.
  - `https://numpy.org/`
- Visit the official website of Matplotlib, and learn about usage of Matplotlib.

  - `https://matplotlib.org/`

- Visit the official website of SciPy, and learn about usage of SciPy.
  - `https://www.scipy.org/`
- Read the textbook "Everything You Wanted to Know About Data Analysis and Fitting but Were Afraid to Ask" and learn about least squares fitting. The Library of NCU has a license for ebook of this textbook, and you can download PDF file.
  - Everything You Wanted to Know About Data Analysis and Fitting but Were Afraid to Ask
    - Peter Young
    - Springer (SpringerBriefs in Physics)
    - `https://www.springer.com/gp/book/9783319190501`

10 Assignment

1. What is dark current? Describe dark current. How to take dark frames? Under which condition, do we need to take dark frames during the observing run?
2. Choose three dark frames. Make your own Python script to visualise those three dark frames using Matplotlib. Show the source code of your Python script and images you have produced.
3. Choose three dark frames. Make your own Python script to make histograms of those three dark 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 dark frames. Make your own Python script to calculate mean, median, variance, standard deviation, minimum value, maximum value of those three dark 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 dark frames. Make your own Python script to calculate mean, median, variance, standard deviation, minimum value, maximum value of those three dark 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. Choose one dark frame. Set four regions of 512×512 pixels on the image. Make your own Python script to calculate mean, median, variance, standard deviation, minimum value, maximum value of those four regions. Discuss the uniformity of the dark frame.
7. Make your own Python script to combine 10 dark frames of 3600-sec exposure time using minmax rejection. Compare combined dark frame with the one combined using sigma clipping. Discuss pros and cons of minmax rejection and sigma clipping. For minmax rejection, you may refer to the IRAF's help.
   - `https://iraf.net/irafhelp.php?val=immatch.imcombine`
8. Make your own Python script which works like `imcombine` task of IRAF (Image Reduction and Analysis Facility). Show the source code of your function. 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 `imcombine` are implemented in your Python script. Mention which functions of `imcombine` are not implemented in your Python script. About `imcombine` task, you may refer to the IRAF's help.
   - `https://iraf.net/irafhelp.php?val=immatch.imcombine`
9. Describe least squares method. What is degree of freedom? What is χ-square? What is reduced χ-square? How to estimate errors of fitted coefficients?
10. Choose one bias frame from the data for this session. Make a histogram of pixel values of the bias frame. Use Gaussian distribution to fit the histogram. Superimpose result of your fitting on the histogram. 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 fitting.

11. Choose one dark frame from the data for this session. Make a histogram of pixel values of the dark frame. Use Gaussian distribution to fit the histogram. Superimpose result of your fitting on the histogram. 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 fitting.