Advanced Astronomical Observations 2022
Session 04: Dark Current

Kinoshita Daisuke

11 March 2022
publicly accessible version

About this file...

- Important information about this file
  - The author of this file is Kinoshita Daisuke.
  - The original version of this file was used for the course "Advanced Astronomical Observations" (course ID: AS6005) offered at Institute of Astronomy, National Central University from February 2022 to June 2022.
  - The file is provided in the hope that it will be useful, but there is no guarantee for the correctness. Use this file at your own risk.
  - If you are willing to use this file for your study, please feel free to use. I'll be very happy to receive feedback from you.
  - If you are willing to use this file for your teaching, please contact to Kinoshita Daisuke. When you use this file partly or entirely, please mention clearly that the author of the original version is Kinoshita Daisuke. Please help me to improve the contents of this file by sending your feedback.
  - Contact address: `https://www.instagram.com/daisuke23888/`

For this session, we examine dark frames. Dark 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. The size of the file is about 350 MB.

- `https://s3b.astro.ncu.edu.tw/advobs_202202/data/data_s04.tar.xz`

Make your own Python script to download the file. Here is an example.

Python Code 1: advobs202202_s04_01.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing re module
import re

# importing pathlib module
import pathlib

# importing ssl module
```

```
import ssl

# importing urllib module
import urllib.request

# setting for SSL
ssl._create_default_https_context = ssl._create_unverified_context

# construction of parser object
desc = 'WWW fetch'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('URL', default='', help='URL of the resource')
parser.add_argument ('-o', '--output', default='', help='output file name')

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

# parameters
target_url  = args.URL
file_output = args.output

#
# check of URL
#
# making a pattern for matching by regular expression
pattern_http = re.compile ('^http')
# matching by regular expression
match_http   = re.search (pattern_http, target_url)
# if not matching, then stop the script
if not (match_http):
    # printing message
    print ("URL has to start with \"http\"!")
    print ("Check the URL!")
    print ("Stopping the script...")
    # exiting the script
    sys.exit ()

# output file name
if (file_output == ''):
    # default output file name
    #   for URL of https://aaa.bbb.ccc/ddd/eee/fff.ggg
    #   output file name ==> fff.ggg
    file_output = target_url.split ('/') [-1]

# existence check of output file
path_output = pathlib.Path (file_output)
if (path_output.exists ()):
    # printing message
    print ("The output file \"%s\" exists!" % file_output)
    print ("Stopping the script...")
    # exiting the script
    sys.exit ()

# printing input parameters
print ("#")
print ("# input parameters")
print ("#  target URL  = %s" % target_url)
```

```
print ("#  output file = %s" % file_output)
print ("#")

# making a request object
req = urllib.request.Request (url=target_url)

# printing status
print ("# now fetching the object...")

# retrieval of target
with urllib.request.urlopen (req) as www:
    # target file size
    file_size_byte = int (www.length)
    # printing file size of target
    print ("#  file size = %10d byte" % (file_size_byte) )
    print ("#            = %10d kB" % (file_size_byte / 1024) )
    print ("#            = %10d MB" % (file_size_byte / 1024 / 1024) )
    # retrieving data
    target_data = www.read ()

# printing status
print ("# finished fetching the object!")

# printing status
print ("# now writing data into file...")

# writing data into output file
with open (file_output, 'wb') as fh:
    fh.write (target_data)

# printing status
print ("# finished writing data into file!")
```

Execute the script as follows.

```
% chmod a+x advobs202202_s04_01.py
% ./advobs202202_s04_01.py -h
usage: advobs202202_s04_01.py [-h] [-o OUTPUT] URL

WWW fetch

positional arguments:
  URL                   URL of the resource

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        output file name

% ./advobs202202_s04_01.py \
? https://s3b.astro.ncu.edu.tw/advobs_202202/data/data_s04.tar.xz
#
# input parameters
#  target URL  = https://s3b.astro.ncu.edu.tw/advobs_202202/data/data_s04.tar.xz
#  output file = data_s04.tar.xz
#
# now fetching the object...
#  file size =  370708536 byte
```

```
#             =     362020 kB
#             =        353 MB
# finished fetching the object!
# now writing data into file...
# finished writing data into file!
% ls -lF
total 354
-rwxr-xr-x  1 daisuke  taiwan       2701 Mar 10 14:19 advobs202202_s04_01.py*
-rw-r--r--  1 daisuke  taiwan         80 Mar 10 13:25 advobs202202_s04_01.py~
-rw-r--r--  1 daisuke  taiwan  370708536 Mar 10 14:26 data_s04.tar.xz
% file data_s04.tar.xz
data_s04.tar.xz: XZ compressed data, checksum CRC64
```

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

```
% curl -k -o data_s04.tar.xz \
? https://s3b.astro.ncu.edu.tw/advobs_202202/data/data_s04.tar.xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  353M  100  353M    0     0  5507k      0  0:01:05  0:01:05 --:--:-- 5477k
% ls -lF data_s04.tar.xz
-rw-r--r--  1 daisuke  taiwan  370708536 Mar 10 14:28 data_s04.tar.xz
```

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_s04.tar.xz
x data_s04/
x data_s04/lot_20210210_1020.fits
x data_s04/lot_20210210_1021.fits
x data_s04/lot_20210210_1022.fits
x data_s04/lot_20210210_1023.fits
x data_s04/lot_20210210_1024.fits
x data_s04/lot_20210210_1025.fits
x data_s04/lot_20210210_1026.fits
x data_s04/lot_20210210_1027.fits
x data_s04/lot_20210210_1028.fits
x data_s04/lot_20210210_1029.fits

.....

x data_s04/lot_20210210_1140.fits
x data_s04/lot_20210210_1141.fits
x data_s04/lot_20210210_1142.fits
x data_s04/lot_20210210_1143.fits
x data_s04/lot_20210210_1144.fits
x data_s04/lot_20210210_1145.fits
x data_s04/lot_20210210_1146.fits
x data_s04/lot_20210210_1147.fits
x data_s04/lot_20210210_1148.fits
x data_s04/lot_20210210_1149.fits
% ls -lF data_s04 | head
```

```
total 1049
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1020.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1021.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1022.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1023.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1024.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1025.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1026.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1027.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1028.fits
% ls data_s04/*.fits | wc
     130     130    4160
```

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

```
% unxz -c data_s04.tar.xz | tar xvf -
x data_s04/
x data_s04/lot_20210210_1020.fits
x data_s04/lot_20210210_1021.fits
x data_s04/lot_20210210_1022.fits
x data_s04/lot_20210210_1023.fits
x data_s04/lot_20210210_1024.fits
x data_s04/lot_20210210_1025.fits
x data_s04/lot_20210210_1026.fits
x data_s04/lot_20210210_1027.fits
x data_s04/lot_20210210_1028.fits
x data_s04/lot_20210210_1029.fits

.....

x data_s04/lot_20210210_1140.fits
x data_s04/lot_20210210_1141.fits
x data_s04/lot_20210210_1142.fits
x data_s04/lot_20210210_1143.fits
x data_s04/lot_20210210_1144.fits
x data_s04/lot_20210210_1145.fits
x data_s04/lot_20210210_1146.fits
x data_s04/lot_20210210_1147.fits
x data_s04/lot_20210210_1148.fits
x data_s04/lot_20210210_1149.fits
% ls -lF data_s04 | head
total 1049
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1020.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1021.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1022.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1023.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1024.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1025.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1026.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1027.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1028.fits
% ls data_s04/*.fits | wc
     130     130    4160
```

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/`

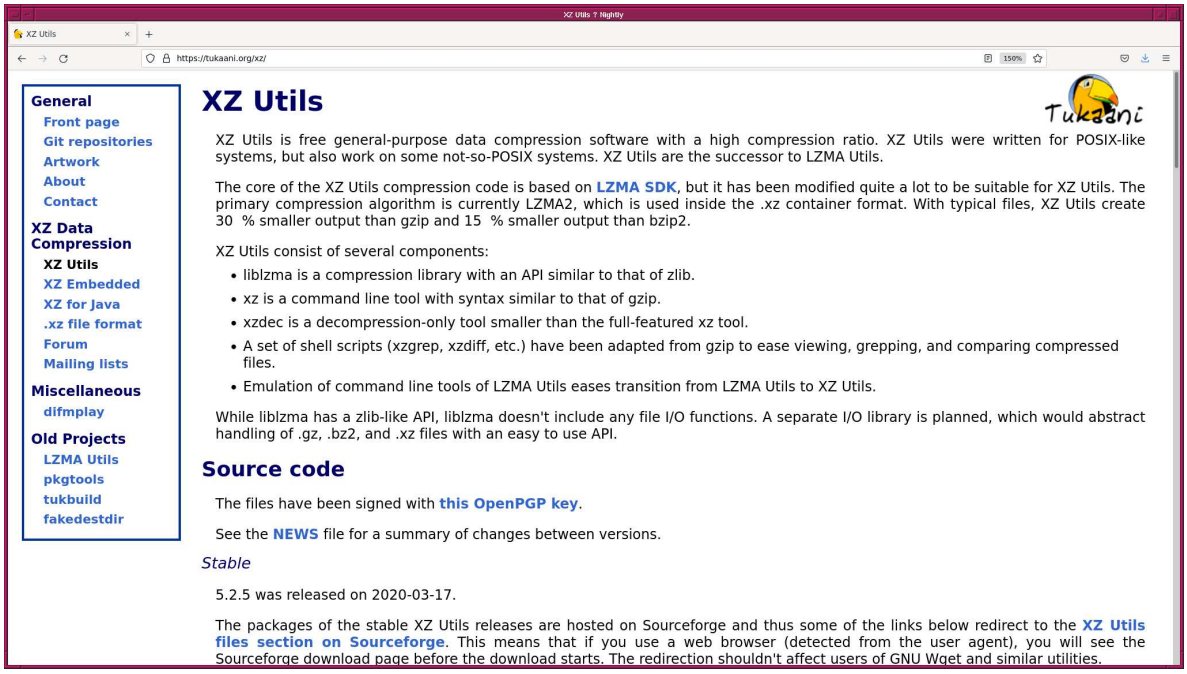

Figure 1: The website of XZ Utils.

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

```
% ls -lF data_s04.tar.xz
-rw-r--r--  1 daisuke  taiwan  370708536 Mar 10 14:28 data_s04.tar.xz
% unxz data_s04.tar.xz
% ls -lF data_s04.tar
-rw-r--r--  1 daisuke  taiwan  1091452416 Mar 10 14:28 data_s04.tar
% file data_s04.tar
data_s04.tar: POSIX tar archive
% tar xvf data_s04.tar
x data_s04/
x data_s04/lot_20210210_1020.fits
x data_s04/lot_20210210_1021.fits
x data_s04/lot_20210210_1022.fits
x data_s04/lot_20210210_1023.fits
x data_s04/lot_20210210_1024.fits
x data_s04/lot_20210210_1025.fits
x data_s04/lot_20210210_1026.fits
x data_s04/lot_20210210_1027.fits
x data_s04/lot_20210210_1028.fits
x data_s04/lot_20210210_1029.fits

.....

x data_s04/lot_20210210_1140.fits
x data_s04/lot_20210210_1141.fits
x data_s04/lot_20210210_1142.fits
x data_s04/lot_20210210_1143.fits
x data_s04/lot_20210210_1144.fits
x data_s04/lot_20210210_1145.fits
x data_s04/lot_20210210_1146.fits
x data_s04/lot_20210210_1147.fits
x data_s04/lot_20210210_1148.fits
x data_s04/lot_20210210_1149.fits
```

```
% ls -lF data_s04 | head
total 1049
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1020.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1021.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1022.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1023.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1024.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1025.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1026.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1027.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 11  2021 lot_20210210_1028.fits
% ls data_s04/*.fits | wc
     130     130    4160
```

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. (Fig. 2)

```
% python3.9
Python 3.9.10 (main, Feb  6 2022, 14:59:28)
[GCC 10.3.0] 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.

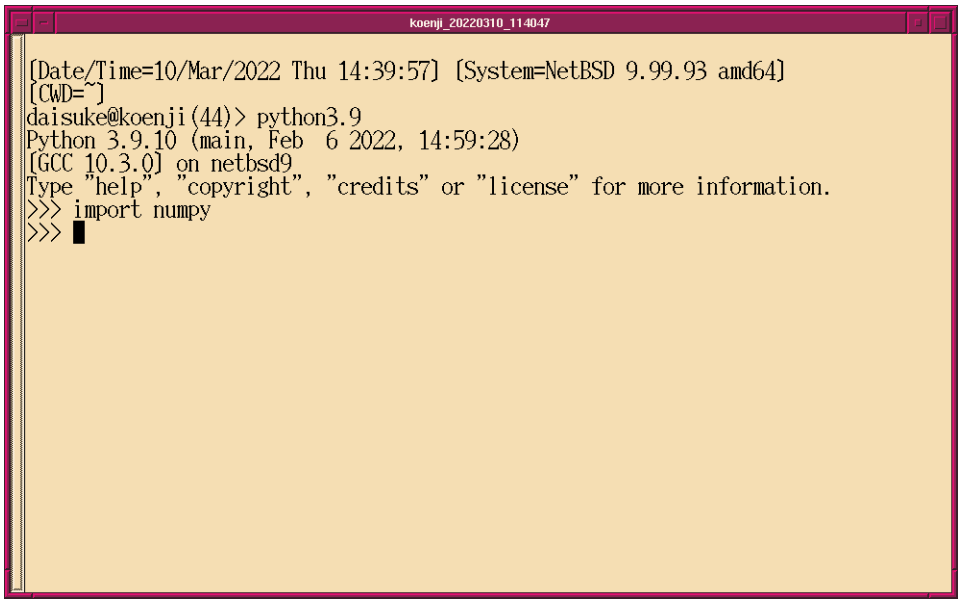

Figure 2: Importing Numpy using interactive mode of Python.

If you do not have Numpy on your computer, then you probably see a message like following.

```
% python3.9
Python 3.9.10 (main, Jan 31 2022, 14:23:57)
[GCC 10.3.0] 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'
>>> exit ()
```

If you do not have Numpy on your computer, then visit the official website of Numpy (Fig. 3). Read the official documentation and install Numpy on your computer.

- https://numpy.org/

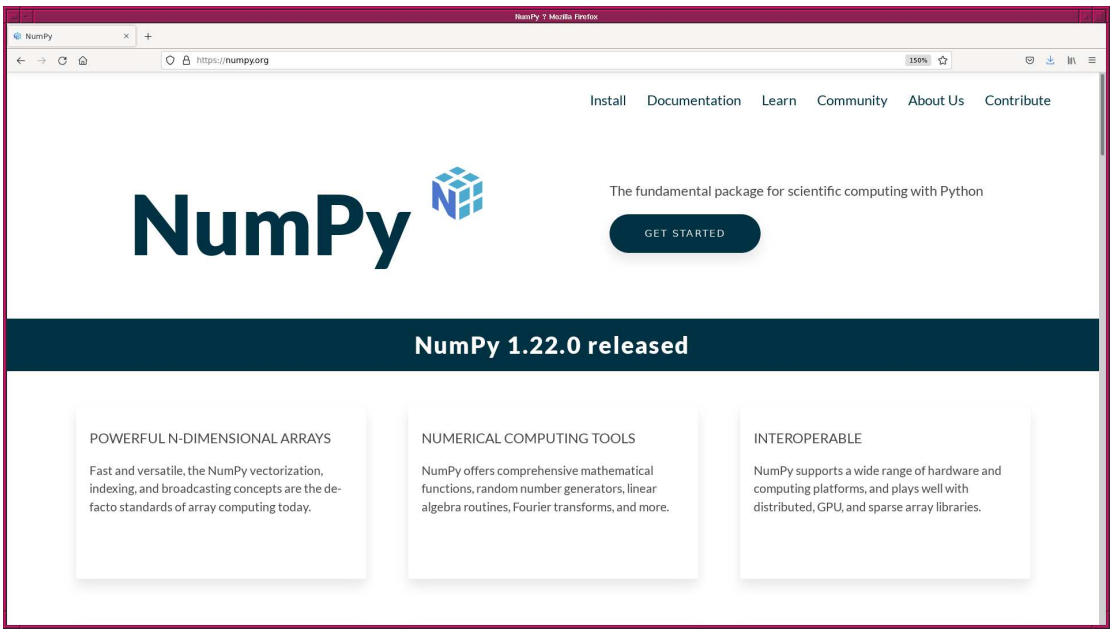

Figure 3: The official website of Numpy.

3.2 Creating a Numpy array

Make a Python script to create a Numpy array.

Python Code 2: advobs202202_s04_02.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 advobs202202_s04_02.py
% ./advobs202202_s04_02.py
[ 1.2  3.4  5.6  7.8  9.  12.3]
```

3.3 Printing information of a Numpy array

Make a Python script to show information of a Numpy array.

Python Code 3: advobs202202_s04_03.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 information
print ("Information of Numpy array:")
print ("  object type        = %s" % type (a) )
print ("  dimensions of data = %s" % a.ndim)
print ("  shape of data      = %s" % str (a.shape) )
print ("  number of elements = %s" % a.size)
print ("  data type          = %s" % a.dtype)
```

Execute the script.

```
% chmod a+x advobs202202_s04_03.py
% ./advobs202202_s04_03.py
Information of Numpy array:
  object type        = <class 'numpy.ndarray'>
  dimensions of data = 1
  shape of data      = (6,)
  number of elements = 6
  data type          = float64
```

3.4 Creating a 2-dim. Numpy array

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

Python Code 4: advobs202202_s04_04.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)

# printing Numpy array information
print ("Information of Numpy array:")
print ("  object type        = %s" % type (b) )
print ("  dimensions of data = %s" % b.ndim)
print ("  shape of data      = %s" % str (b.shape) )
print ("  number of elements = %s" % b.size)
print ("  data type          = %s" % b.dtype)
```

Execute the script.

```
% chmod a+x advobs202202_s04_04.py
% ./advobs202202_s04_04.py
[[ 1.2  3.4  5.6]
 [ 7.8  9.  12.3]
 [ 1.   2.   3. ]]
Information of Numpy array:
  object type        = <class 'numpy.ndarray'>
  dimensions of data = 2
  shape of data      = (3, 3)
  number of elements = 9
  data type          = float64
```

3.5 Arithmetic operations of 2-dim. Numpy arrays

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

Python Code 5: advobs202202_s04_05.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 arrays
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 a+x advobs202202_s04_05.py
% ./advobs202202_s04_05.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.        ]]
```

3.6 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 using a function `numpy.concatenate`.

Python Code 6: advobs202202_s04_06.py

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

# importing Numpy module
import numpy
import numpy.random

#
# parameters
#

# number of elements in X-axis and Y-axis
n_x   = 5
n_y   = 5

# mean and stddev for random number generation
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)

# printing cube
print ("cube =")
print (cube)

# 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 advobs202202_s04_06.py
% ./advobs202202_s04_06.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)
cube =
[[[69.70641894 47.40613203 57.05148909 45.84734552 41.48170517]
  [25.84842179 66.42556098 47.03382366 44.74362025 65.88124115]
  [35.14060875 58.61345415 37.98266659 58.5471095  54.1348298 ]
  [50.47609567 43.03488238 45.24691699 50.14953111 37.47164824]
  [60.80762791 57.07571756 57.90604784 57.48066974 35.20782881]]

 [[58.79519893 51.65458399 69.65261756 37.64973114 29.34172581]
  [62.54602489 49.23195616 61.217034   41.576212   59.04513789]
```

```
  [60.13101368 35.4619919  59.58776604 37.1111063  44.72345708]
  [46.80248933 50.58433106 27.54039326 34.91038184 41.40852443]
  [35.17755234 52.13370127 49.01265786 50.72897962 50.8158284 ]]

 [[54.17300688 49.9061189  54.30911082 52.78999035 54.183686  ]
  [47.19624728 53.76397518 58.98673614 81.99618556 37.67376813]
  [46.16401519 54.18266814 46.572561   55.36646366 40.24343734]
  [44.05594978 62.10427703 56.93817443 51.58592355 56.40292802]
  [59.86264443 66.16296071 49.45950097 51.12721012 51.83404387]]]
shape of cube2 = (5, 5, 5)
cube2 =
[[[69.70641894 47.40613203 57.05148909 45.84734552 41.48170517]
  [25.84842179 66.42556098 47.03382366 44.74362025 65.88124115]
  [35.14060875 58.61345415 37.98266659 58.5471095  54.1348298 ]
  [50.47609567 43.03488238 45.24691699 50.14953111 37.47164824]
  [60.80762791 57.07571756 57.90604784 57.48066974 35.20782881]]

 [[58.79519893 51.65458399 69.65261756 37.64973114 29.34172581]
  [62.54602489 49.23195616 61.217034   41.576212   59.04513789]
  [60.13101368 35.4619919  59.58776604 37.1111063  44.72345708]
  [46.80248933 50.58433106 27.54039326 34.91038184 41.40852443]
  [35.17755234 52.13370127 49.01265786 50.72897962 50.8158284 ]]

 [[54.17300688 49.9061189  54.30911082 52.78999035 54.183686  ]
  [47.19624728 53.76397518 58.98673614 81.99618556 37.67376813]
  [46.16401519 54.18266814 46.572561   55.36646366 40.24343734]
  [44.05594978 62.10427703 56.93817443 51.58592355 56.40292802]
  [59.86264443 66.16296071 49.45950097 51.12721012 51.83404387]]

 [[56.08280202 49.2360081  56.3963364  62.39639802 36.52384714]
  [40.73694993 51.40680341 62.1569097  56.58400722 52.84313599]
  [52.23363801 44.13215296 58.2375673  64.00010771 44.78194775]
  [47.33510697 60.07558987 49.03693986 50.61275641 60.56588183]
  [49.84960201 42.36997064 56.20393549 49.51465757 67.8187214 ]]

 [[48.15312501 50.77166866 60.03467304 50.43332744 50.99341806]
  [60.76158091 31.97035284 48.99675682 77.08983152 43.02063986]
  [57.86745362 33.53040742 63.04332183 49.22407483 42.45257096]
  [44.89920455 31.06919185 42.86066223 56.67796394 49.49181876]
  [51.66845954 54.37051074 48.65127331 61.43311934 44.76384192]]]
```

To learn more about the function `numpy.concatenate`, visit following pages and read the documentation. (Fig. 4 and 5)

- `https://numpy.org/doc/stable/user/absolute_beginners.html#adding-removing-and-sorting-elements`
- `https://numpy.org/doc/stable/reference/generated/numpy.concatenate.html`

3.7 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 7: advobs202202_s04_07.py

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

# importing Numpy module
import numpy
```

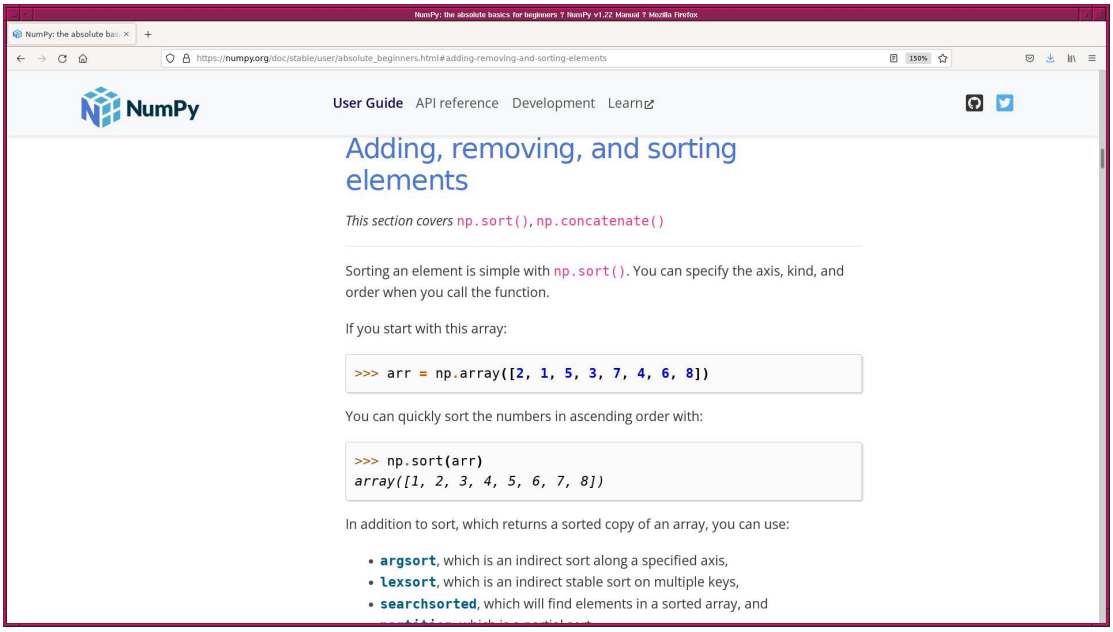

Figure 4: The official document of Numpy "NumPy: the absolute basics for beginners".

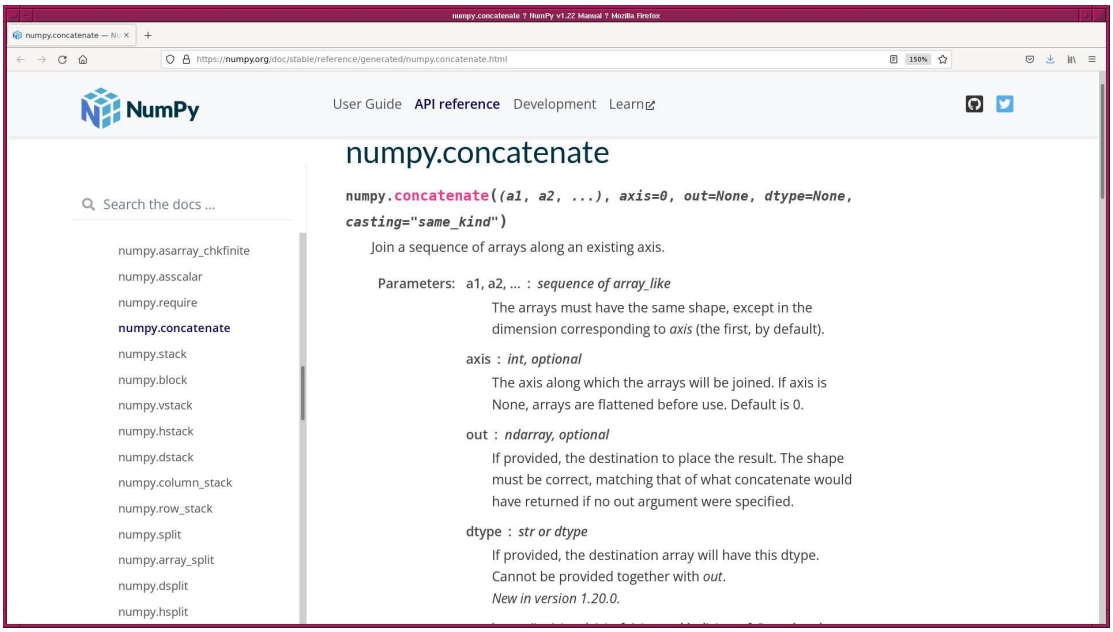

Figure 5: The API reference document of Numpy for the function `concatenate`.

```
import numpy.random

#
# parameters
#

# numbers of elements in X-axis, Y-axis, and Z-axis
n_x    = 256
n_y    = 256
n_z    = 100

# mean and stddev for random number generation
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):
        # for the first 2-dim. array, copy it to "tmp0"
        tmp0 = tmp
    elif (i == 1):
        # for the second 2-dim. array, make a 3-dim. array "cube"
        # by concatenating "tmp0" and "tmp" using the function concatenate
        cube = numpy.concatenate ( ([tmp0], [tmp]), axis=0 )
    else:
        # for other 2-dim. arrays, concatenate "tmp"
        # to the 3-dim. array "cube"
        cube = numpy.concatenate ( (cube, [tmp]), axis=0 )

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

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

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

# printing "combined"
print ("array \"combined\":")
print (combined)
```

Execute the script.

```
% chmod a+x advobs202202_s04_07.py
% ./advobs202202_s04_07.py
shape of array "cube"     = (100, 256, 256)
shape of array "combined" = (256, 256)
array "combined":
[[50.59525963 49.41604405 48.66941678 ... 49.37498088 50.07538668
  51.45625316]
 [49.31256109 50.96312047 48.57289787 ... 50.77241168 50.43035409
  52.01140689]
 [51.6310732  50.63685089 49.37656295 ... 52.01418358 49.62849851
  51.03051201]
```

```
...
[49.48255209 50.16015828 47.94239423 ... 48.55068167 50.29132091
 48.35185604]
[50.26190705 50.63859157 50.90062701 ... 51.04096272 49.89446547
 50.0749644 ]
[49.53511485 49.76267651 50.42280196 ... 50.0000074  51.55566992
 51.04963815]]
```

To learn more about the function `numpy.mean`, visit following web page and read the documentation. (Fig. 6)

- `https://numpy.org/doc/stable/reference/generated/numpy.mean.html`

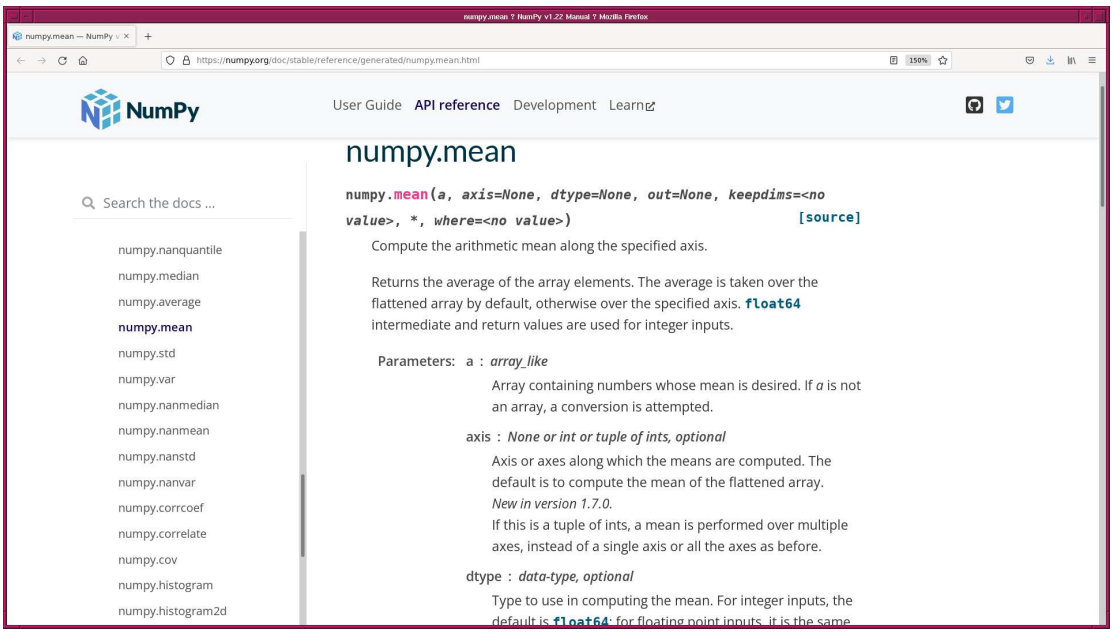

Figure 6: The API reference for the function `numpy.mean`.

3.8 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 8: advobs202202_s04_08.py

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

# importing Numpy module
import numpy
import numpy.random

# importing Astropy module
import astropy.stats

# number of elements in X-axis, Y-axis, and Z-axis
n_x = 256
n_y = 256
n_z = 100

# mean and stddev for random number generation
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) )

    # choosing a pixel for an outlier
    x = int ( numpy.random.uniform (0, n_x) )
    y = int ( numpy.random.uniform (0, n_y) )
    # adding an outlier of value around 10,000
    tmp[x][y] += 10000.0

    # concatenating 2-dim. arrays to make a data cube
    if (i == 0):
        # for the first 2-dim. array, copy it to "tmp0"
        tmp0 = tmp
    elif (i == 1):
        # for the second 2-dim. array, make a 3-dim. array "cube"
        # by concatenating "tmp0" and "tmp" using the function concatenate
        cube = numpy.concatenate ( ([tmp0], [tmp]), axis=0 )
    else:
        # for other 2-dim. arrays, concatenate "tmp"
        # to the 3-dim. array "cube"
        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
#   threshold = mean +/- 3.0 times of stddev
#   max number of iterations = 10
#   calculation of average = mean
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 advobs202202_s04_08.py
% ./advobs202202_s04_08.py
shape of cube = (100, 256, 256)
```

```
shape of combined = (256, 256)
[[51.57720931 48.00546339 50.19974879 ... 48.81542675 49.38929457
  48.79940065]
 [48.55096736 49.10920667 49.88122711 ... 48.98484588 50.56825624
  48.65816524]
 [48.76614438 50.34881993 50.7693664  ... 51.5066679  49.52795417
  49.24138267]
 ...
 [51.43550582 50.29476188 51.47497344 ... 50.62973788 51.91967566
  50.26587778]
 [49.12146901 49.47743304 52.0183559  ... 49.65931428 49.52943214
  48.05461411]
 [50.60146533 50.20778568 48.62174192 ... 49.11476333 51.28168199
  48.65694975]]
min of combined_simple:  45.866322
max of combined_simple:  153.133449
min of combined_sigclip: 45.866322
max of combined_sigclip: 54.790683
```

To learn more about the function `astropy.stats.sigma_clipped_stats`, visit following web page and read the documentation. (Fig. 7)

- `https://docs.astropy.org/en/stable/api/astropy.stats.sigma_clipped_stats.html`

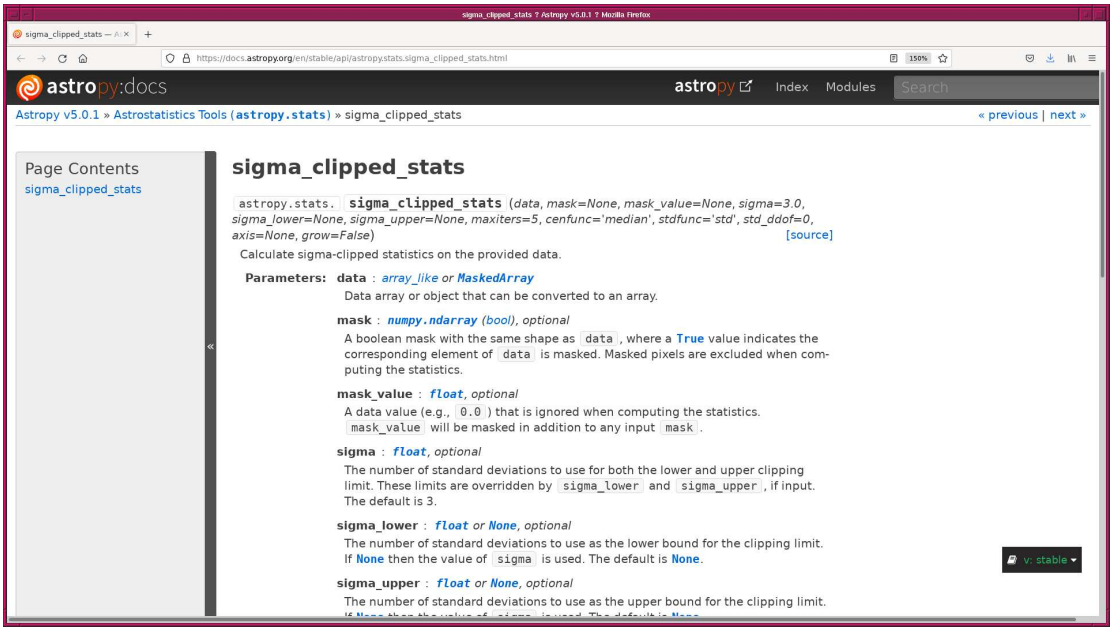

Figure 7: The documentation for the function `astropy.stats.sigma_clipped_stats` of Astropy.

4 Checking data

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

Python Code 9: advobs202202_s04_09.py

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

# importing argparse module
```

```
import argparse

# importing pathlib module
import pathlib

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

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

# input parameters
keywords   = args.keywords
list_files = args.files

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

# printing header
print ("# filename,", keywords)

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

    # making a pathlib object
    path_fits = pathlib.Path (file_fits)

    # if the file does not exist, then skip
    if not (path_fits.exists ()):
        continue

    # file name
    filename = path_fits.name

    # 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 ()
```

```
    # gathering information from FITS header
    record = "%s      " % filename
    # processing for each keyword
    for key in list_keywords:
        if key in header0:
            # if keyword exists, then copy value to the variable "value"
            value = str (header0[key])
        else:
            # if keyword does not exists, then copy "__NONE__"
            # to the varialble "value"
            value = "__NONE__"
        # appending "value" to the end of "record"
        record += " %-8s" % value

    # printing information
    print (record)
```

Execute the script and generate a simple observing log.

```
% chmod a+x advobs202202_s04_09.py
% ./advobs202202_s04_09.py -h
usage: advobs202202_s04_09.py [-h] [-k KEYWORDS] files [files ...]

Generating a simple observing log

positional arguments:
  files                 FITS files

optional arguments:
  -h, --help            show this help message and exit
  -k KEYWORDS, --keywords KEYWORDS
                        a list of keyword to check (e.g. TIME-OBS,EXPTIME)

% ./advobs202202_s04_09.py data_s04/*
# filename, TIME-OBS,IMAGETYP,EXPTIME,FILTER
lot_20210210_1020.fits       20:07:18 BIAS     0.0      __NONE__
lot_20210210_1021.fits       20:07:23 BIAS     0.0      __NONE__
lot_20210210_1022.fits       20:07:28 BIAS     0.0      __NONE__
lot_20210210_1023.fits       20:07:33 BIAS     0.0      __NONE__
lot_20210210_1024.fits       20:07:38 BIAS     0.0      __NONE__
lot_20210210_1025.fits       20:07:43 BIAS     0.0      __NONE__
lot_20210210_1026.fits       20:07:48 BIAS     0.0      __NONE__
lot_20210210_1027.fits       20:07:53 BIAS     0.0      __NONE__
lot_20210210_1028.fits       20:07:58 BIAS     0.0      __NONE__
lot_20210210_1029.fits       20:08:03 BIAS     0.0      __NONE__
lot_20210210_1030.fits       20:08:07 DARK     10.0     __NONE__
lot_20210210_1031.fits       20:08:22 DARK     10.0     __NONE__
lot_20210210_1032.fits       20:08:37 DARK     10.0     __NONE__
lot_20210210_1033.fits       20:08:52 DARK     10.0     __NONE__
lot_20210210_1034.fits       20:09:07 DARK     10.0     __NONE__
lot_20210210_1035.fits       20:09:22 DARK     10.0     __NONE__
lot_20210210_1036.fits       20:09:37 DARK     10.0     __NONE__
lot_20210210_1037.fits       20:09:52 DARK     10.0     __NONE__
lot_20210210_1038.fits       20:10:07 DARK     10.0     __NONE__
lot_20210210_1039.fits       20:10:22 DARK     10.0     __NONE__
lot_20210210_1040.fits       20:10:37 BIAS     0.0      __NONE__
lot_20210210_1041.fits       20:10:42 BIAS     0.0      __NONE__
lot_20210210_1042.fits       20:10:47 BIAS     0.0      __NONE__
```

```
lot_20210210_1043.fits      20:10:52 BIAS     0.0      __NONE__
lot_20210210_1044.fits      20:10:57 BIAS     0.0      __NONE__
lot_20210210_1045.fits      20:11:02 BIAS     0.0      __NONE__
lot_20210210_1046.fits      20:11:07 BIAS     0.0      __NONE__
lot_20210210_1047.fits      20:11:12 BIAS     0.0      __NONE__
lot_20210210_1048.fits      20:11:17 BIAS     0.0      __NONE__
lot_20210210_1049.fits      20:11:21 BIAS     0.0      __NONE__
lot_20210210_1050.fits      20:11:26 DARK     30.0     __NONE__
lot_20210210_1051.fits      20:12:01 DARK     30.0     __NONE__
lot_20210210_1052.fits      20:12:36 DARK     30.0     __NONE__
lot_20210210_1053.fits      20:13:11 DARK     30.0     __NONE__
lot_20210210_1054.fits      20:13:46 DARK     30.0     __NONE__
lot_20210210_1055.fits      20:14:21 DARK     30.0     __NONE__
lot_20210210_1056.fits      20:14:56 DARK     30.0     __NONE__
lot_20210210_1057.fits      20:15:33 DARK     30.0     __NONE__
lot_20210210_1058.fits      20:16:08 DARK     30.0     __NONE__
lot_20210210_1059.fits      20:16:45 DARK     30.0     __NONE__
lot_20210210_1060.fits      20:17:20 BIAS     0.0      __NONE__
lot_20210210_1061.fits      20:17:25 BIAS     0.0      __NONE__
lot_20210210_1062.fits      20:17:30 BIAS     0.0      __NONE__
lot_20210210_1063.fits      20:17:35 BIAS     0.0      __NONE__
lot_20210210_1064.fits      20:17:40 BIAS     0.0      __NONE__
lot_20210210_1065.fits      20:17:45 BIAS     0.0      __NONE__
lot_20210210_1066.fits      20:17:50 BIAS     0.0      __NONE__
lot_20210210_1067.fits      20:17:55 BIAS     0.0      __NONE__
lot_20210210_1068.fits      20:18:00 BIAS     0.0      __NONE__
lot_20210210_1069.fits      20:18:05 BIAS     0.0      __NONE__
lot_20210210_1070.fits      20:18:10 DARK     90.0     __NONE__
lot_20210210_1071.fits      20:19:45 DARK     90.0     __NONE__
lot_20210210_1072.fits      20:21:19 DARK     90.0     __NONE__
lot_20210210_1073.fits      20:22:54 DARK     90.0     __NONE__
lot_20210210_1074.fits      20:24:29 DARK     90.0     __NONE__
lot_20210210_1075.fits      20:26:04 DARK     90.0     __NONE__
lot_20210210_1076.fits      20:27:39 DARK     90.0     __NONE__
lot_20210210_1077.fits      20:29:14 DARK     90.0     __NONE__
lot_20210210_1078.fits      20:30:49 DARK     90.0     __NONE__
lot_20210210_1079.fits      20:32:24 DARK     90.0     __NONE__
lot_20210210_1080.fits      20:33:59 BIAS     0.0      __NONE__
lot_20210210_1081.fits      20:34:04 BIAS     0.0      __NONE__
lot_20210210_1082.fits      20:34:09 BIAS     0.0      __NONE__
lot_20210210_1083.fits      20:34:14 BIAS     0.0      __NONE__
lot_20210210_1084.fits      20:34:19 BIAS     0.0      __NONE__
lot_20210210_1085.fits      20:34:24 BIAS     0.0      __NONE__
lot_20210210_1086.fits      20:34:29 BIAS     0.0      __NONE__
lot_20210210_1087.fits      20:34:34 BIAS     0.0      __NONE__
lot_20210210_1088.fits      20:34:39 BIAS     0.0      __NONE__
lot_20210210_1089.fits      20:34:44 BIAS     0.0      __NONE__
lot_20210210_1090.fits      20:34:49 DARK     300.0    __NONE__
lot_20210210_1091.fits      20:39:54 DARK     300.0    __NONE__
lot_20210210_1092.fits      20:44:59 DARK     300.0    __NONE__
lot_20210210_1093.fits      20:50:04 DARK     300.0    __NONE__
lot_20210210_1094.fits      20:55:09 DARK     300.0    __NONE__
lot_20210210_1095.fits      21:00:14 DARK     300.0    __NONE__
lot_20210210_1096.fits      21:05:19 DARK     300.0    __NONE__
lot_20210210_1097.fits      21:10:24 DARK     300.0    __NONE__
lot_20210210_1098.fits      21:15:29 DARK     300.0    __NONE__
lot_20210210_1099.fits      21:20:34 DARK     300.0    __NONE__
lot_20210210_1100.fits      21:25:39 BIAS     0.0      __NONE__
lot_20210210_1101.fits      21:25:44 BIAS     0.0      __NONE__
```

```
lot_20210210_1102.fits        21:25:49 BIAS     0.0       __NONE__
lot_20210210_1103.fits        21:25:54 BIAS     0.0       __NONE__
lot_20210210_1104.fits        21:25:59 BIAS     0.0       __NONE__
lot_20210210_1105.fits        21:26:04 BIAS     0.0       __NONE__
lot_20210210_1106.fits        21:26:09 BIAS     0.0       __NONE__
lot_20210210_1107.fits        21:26:14 BIAS     0.0       __NONE__
lot_20210210_1108.fits        21:26:19 BIAS     0.0       __NONE__
lot_20210210_1109.fits        21:26:24 BIAS     0.0       __NONE__
lot_20210210_1110.fits        21:26:29 DARK     900.0     __NONE__
lot_20210210_1111.fits        21:41:34 DARK     900.0     __NONE__
lot_20210210_1112.fits        21:56:39 DARK     900.0     __NONE__
lot_20210210_1113.fits        22:11:43 DARK     900.0     __NONE__
lot_20210210_1114.fits        22:26:48 DARK     900.0     __NONE__
lot_20210210_1115.fits        22:41:53 DARK     900.0     __NONE__
lot_20210210_1116.fits        22:56:58 DARK     900.0     __NONE__
lot_20210210_1117.fits        23:12:03 DARK     900.0     __NONE__
lot_20210210_1118.fits        23:27:08 DARK     900.0     __NONE__
lot_20210210_1119.fits        23:42:13 DARK     900.0     __NONE__
lot_20210210_1120.fits        23:57:18 BIAS     0.0       __NONE__
lot_20210210_1121.fits        23:57:23 BIAS     0.0       __NONE__
lot_20210210_1122.fits        23:57:28 BIAS     0.0       __NONE__
lot_20210210_1123.fits        23:57:33 BIAS     0.0       __NONE__
lot_20210210_1124.fits        23:57:38 BIAS     0.0       __NONE__
lot_20210210_1125.fits        23:57:43 BIAS     0.0       __NONE__
lot_20210210_1126.fits        23:57:48 BIAS     0.0       __NONE__
lot_20210210_1127.fits        23:57:53 BIAS     0.0       __NONE__
lot_20210210_1128.fits        23:57:58 BIAS     0.0       __NONE__
lot_20210210_1129.fits        23:58:03 BIAS     0.0       __NONE__
lot_20210210_1130.fits        23:58:08 DARK     3600.0    __NONE__
lot_20210210_1131.fits        00:58:13 DARK     3600.0    __NONE__
lot_20210210_1132.fits        01:58:25 DARK     3600.0    __NONE__
lot_20210210_1133.fits        02:58:30 DARK     3600.0    __NONE__
lot_20210210_1134.fits        03:58:42 DARK     3600.0    __NONE__
lot_20210210_1135.fits        04:58:47 DARK     3600.0    __NONE__
lot_20210210_1136.fits        05:58:52 DARK     3600.0    __NONE__
lot_20210210_1137.fits        06:58:57 DARK     3600.0    __NONE__
lot_20210210_1138.fits        07:59:03 DARK     3600.0    __NONE__
lot_20210210_1139.fits        08:59:08 DARK     3600.0    __NONE__
lot_20210210_1140.fits        09:59:13 BIAS     0.0       __NONE__
lot_20210210_1141.fits        09:59:21 BIAS     0.0       __NONE__
lot_20210210_1142.fits        09:59:26 BIAS     0.0       __NONE__
lot_20210210_1143.fits        09:59:31 BIAS     0.0       __NONE__
lot_20210210_1144.fits        09:59:36 BIAS     0.0       __NONE__
lot_20210210_1145.fits        09:59:41 BIAS     0.0       __NONE__
lot_20210210_1146.fits        09:59:46 BIAS     0.0       __NONE__
lot_20210210_1147.fits        09:59:51 BIAS     0.0       __NONE__
lot_20210210_1148.fits        09:59:56 BIAS     0.0       __NONE__
lot_20210210_1149.fits        10:00:01 BIAS     0.0       __NONE__
```

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 10: advobs202202_s04_08.py

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

```
# importing argparse module
import argparse

# importing pathlib module
import pathlib

# 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 using SciPy'
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 (default: none)')
parser.add_argument ('-t', '--threshold', type=float, default=4.0, \
                     help='rejection threshold in sigma (default: 4.0)')
parser.add_argument ('files', nargs='+', help='FITS files')

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

# input parameters
list_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_fits in list_files:
    # if the extension of the file is not '.fits', then skip
    if (file_fits[-5:] != '.fits'):
        continue

    # if the file does not exist, then skip
    path_fits = pathlib.Path (file_fits)
    if not (path_fits.exists ()):
        continue

    # file name
    filename = path_fits.name

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

```
    # 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 (2-dim. data --> 1-dim. 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 advobs202202_s04_10.py
% ./advobs202202_s04_10.py -h
usage: advobs202202_s04_10.py [-h] [-r {none,sigclip}] [-t THRESHOLD]
                              files [files ...]

Calculating statistical values using SciPy

positional arguments:
  files                 FITS files

optional arguments:
  -h, --help            show this help message and exit
  -r {none,sigclip}, --rejection {none,sigclip}
                        outlier rejection algorithm (default: none)
  -t THRESHOLD, --threshold THRESHOLD
                        rejection threshold in sigma (default: 4.0)

% ./advobs202202_s04_10.py data_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 sigma clipping to reject pixels with cosmic ray hits.

```
% ./advobs202202_s04_10.py -r sigclip data_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
--------------------------------------------------------------------------------
```

To learn more about the function `scipy.stats.sigmaclip`, visit following web page and read the documentation. (Fig. 8)

- `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.sigmaclip.html`

To learn more about functions `numpy.nanmean`, `numpy.nanmedian`, `numpy.nanstd`, `numpy.nanmin`, and `numpy.nanmax`, visit following web pages and read documentation.

- `https://numpy.org/doc/stable/reference/generated/numpy.nanmean.html` (Fig. 9)
- `https://numpy.org/doc/stable/reference/generated/numpy.nanmedian.html`
- `https://numpy.org/doc/stable/reference/generated/numpy.nanstd.html`
- `https://numpy.org/doc/stable/reference/generated/numpy.nanmin.html`
- `https://numpy.org/doc/stable/reference/generated/numpy.nanmax.html`

Here is a comparison of `numpy.mean` and `numpy.nanmean`. If NaN is included in a Numpy array, `numpy.mean` does not give a result that we expect.

Python Code 11: advobs202202_s04_11.py

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

# importing numpy module
import numpy
```

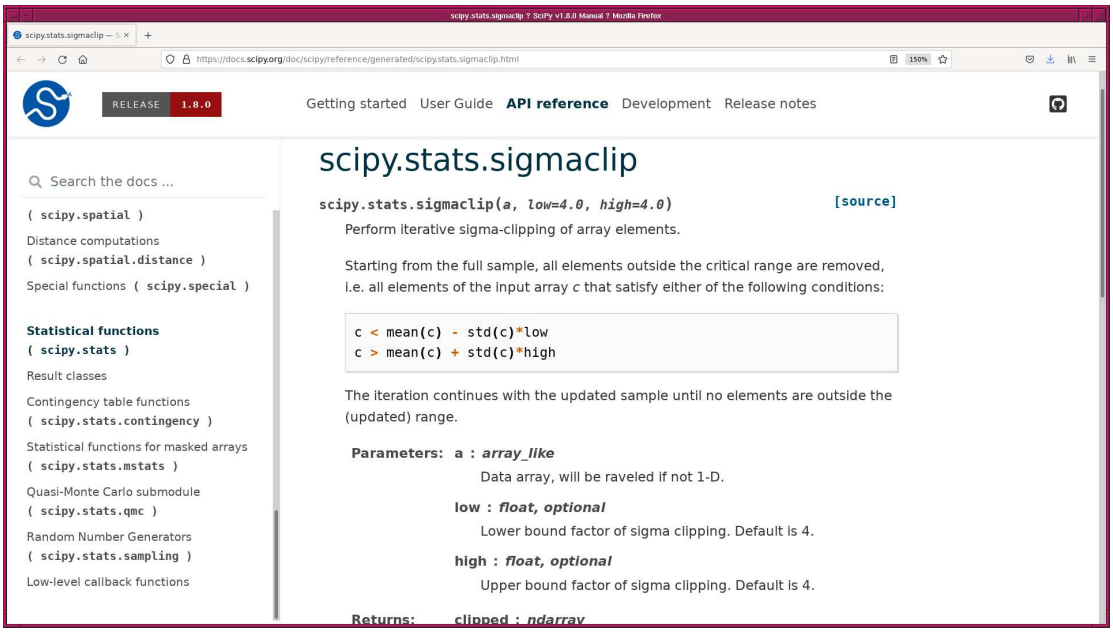

Figure 8: The official documentation of the function `scipy.stats.sigmaclip` of SciPy.

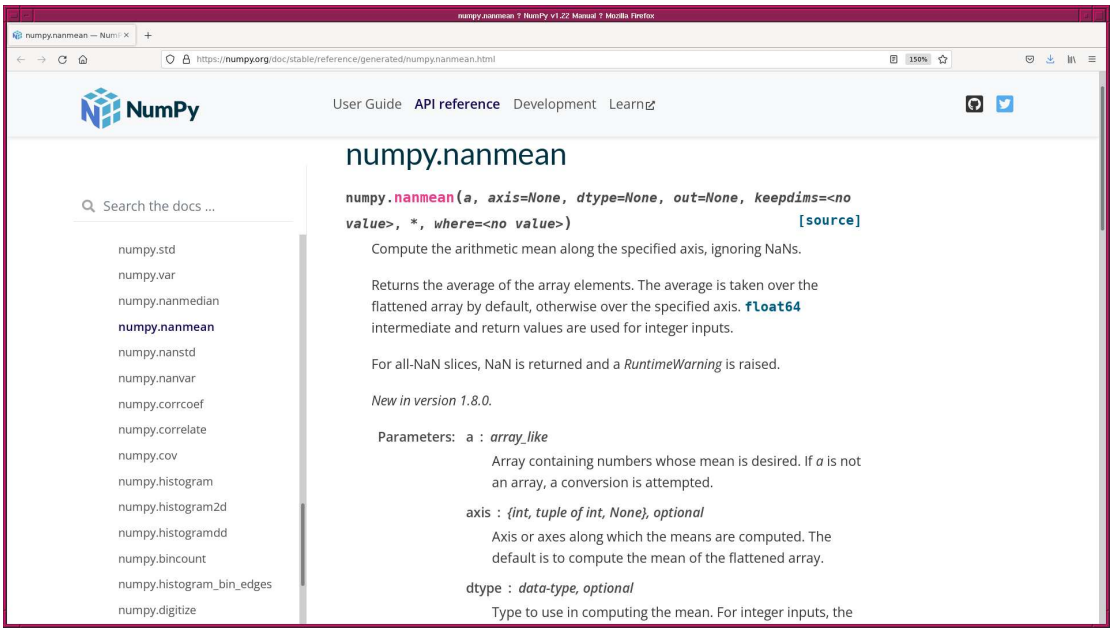

Figure 9: The official documentation of the function `numpy.nanmean` of Numpy.

```
# sample data set 1
data1 = numpy.array ([10.0, 10.0, 9.0, 9.0, 11.0, 11.0])

# calculation of simple mean
data1_mean = numpy.mean (data1)

# printing result
print ("data1         =", data1)
print ("data1_mean    = %f" % data1_mean)

# sample data set 2 with NaN value
data2 = numpy.array ([10.0, 10.0, 9.0, 9.0, 11.0, 11.0, numpy.nan])

# calculation of simple mean
data2_mean = numpy.mean (data2)

# printing result
print ("data2         =", data2)
print ("data2_mean    = %f" % data2_mean)

# calculation of simple mean using numpy.nanmean
data2_nanmean = numpy.nanmean (data2)

# printing result
print ("data2_nanmean = %f" % data2_nanmean)
```

Execute the script.

```
% chmod a+x advobs202202_s04_11.py
% ./advobs202202_s04_11.py
data1         = [10. 10.  9.  9. 11. 11.]
data1_mean    = 10.000000
data2         = [10. 10.  9.  9. 11. 11. nan]
data2_mean    = nan
data2_nanmean = 10.000000
```

6 Visualising a dark frame

Make a Python script to visualise a dark frame.

Python Code 12: advobs202202_s04_12.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

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

```
# 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')
parser.add_argument ('-r', '--resolution', type=int, default=450, \
                     help='resolution of output file (default: 450 dpi)')

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

# input FITS file
file_input  = args.input
file_output = args.output
resolution  = args.resolution

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

# input file existence check
path_input = pathlib.Path (file_input)
if not (path_input.exists ()):
    # printing a message
    print ("Error: the file \"%s\" does not exists!" % file_input)
    # exit
    sys.exit ()

# output file existence check
path_output = pathlib.Path (file_output)
if (path_output.exists ()):
    # printing a message
    print ("Error: the file \"%s\" exists!" % file_output)
    # exit
    sys.exit ()

# printing input parameters
print ("#")
print ("# Input parameters:")
print ("#   input file  = %s" % file_input)
```

```
print ("#   output file = %s" % file_output)
print ("#   resolution of output image = %d dpi" % resolution)
print ("#")

# printing status
print ("# now, reading a FITS file...")

# 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 ()

# printing status
print ("# finished reading a FITS file!")

# printing status
print ("# now, writing an image file...")

# 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=resolution)

# printing status
print ("# finished writing an image file!")
```

Run the script and generate a PNG file.

```
% chmod a+x advobs202202_s04_12.py
% ./advobs202202_s04_12.py -h
usage: advobs202202_s04_12.py [-h] [-i INPUT] [-o OUTPUT] [-r RESOLUTION]

Reading a FITS file and making a PNG file

optional arguments:
```

```
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        intput FITS file
  -o OUTPUT, --output OUTPUT
                        output image file
  -r RESOLUTION, --resolution RESOLUTION
                        resolution of output file (default: 450 dpi)

% ./advobs202202_s04_12.py -i data_s04/lot_20210210_1130.fits -o 1130.png
#
# Input parameters:
#   input file  = data_s04/lot_20210210_1130.fits
#   output file = 1130.png
#   resolution of output image = 450 dpi
#
# now, reading a FITS file...
# finished reading a FITS file!
# now, writing an image file...
# data_s04/lot_20210210_1130.fits ==> 1130.png
# finished writing an image file!
% ls -lF 1130.png
-rw-r--r--  1 daisuke  taiwan  182545 Mar 10 22:55 1130.png
```

Show the image. (Fig. 10) Because of some pixels with large pixel values, we cannot see the detailed structure of the image.

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

Calculate statistical values of the file `lot_20210210_1130.fits`.

```
% ./advobs202202_s04_10.py -r sigclip data_s04/lot_20210210_1130.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
-------------------------------------------------------------------------------
```

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

Python Code 13: advobs202202_s04_13.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing astropy module
import astropy.io.fits

# importing matplotlib module
```

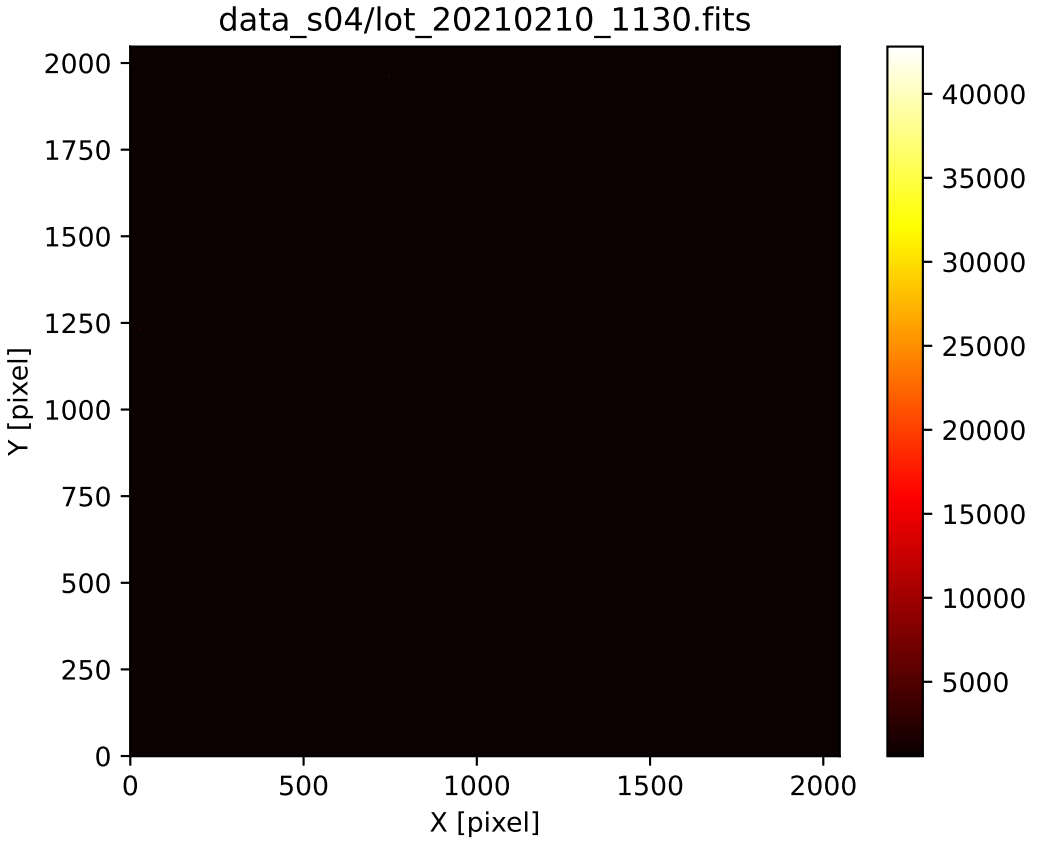

Figure 10: The image of the file `lot_20210210_1130.fits`.

```
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')
parser.add_argument ('-r', '--resolution', type=int, default=450, \
                     help='resolution of output file (default: 450 dpi)')

# 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
resolution  = args.resolution

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

# input file existence check
path_input = pathlib.Path (file_input)
if not (path_input.exists ()):
    # printing a message
    print ("Error: the file \"%s\" does not exists!" % file_input)
    # exit
    sys.exit ()

# output file existence check
path_output = pathlib.Path (file_output)
if (path_output.exists ()):
    # printing a message
    print ("Error: the file \"%s\" exists!" % file_output)
    # exit
    sys.exit ()
```

```
# printing input parameters
print ("#")
print ("# Input parameters:")
print ("#   input file  = %s" % file_input)
print ("#   output file = %s" % file_output)
print ("#   resolution of output image = %d dpi" % resolution)
print ("#   minimum value to plot      = %f" % vmin)
print ("#   maximum value to plot      = %f" % vmax)
print ("#")

# printing status
print ("# now, reading a FITS file...")

# 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 ()

# printing status
print ("# finished reading a FITS file!")

# printing status
print ("# now, writing an image file...")

# 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=resolution)

# printing status
print ("# finished writing an image file!")
```

Run the script and generate a PNG file.

```
% chmod a+x advobs202202_s04_13.py
% ./advobs202202_s04_13.py -h
usage: advobs202202_s04_13.py [-h] [-a MIN] [-b MAX] [-i INPUT] [-o OUTPUT]
                              [-r RESOLUTION]

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
  -i INPUT, --input INPUT
                        intput FITS file
  -o OUTPUT, --output OUTPUT
                        output image file
  -r RESOLUTION, --resolution RESOLUTION
                        resolution of output file (default: 450 dpi)

% ./advobs202202_s04_13.py -i data_s04/lot_20210210_1130.fits -o 1130_2.png \
? -a 550 -b 650
#
# Input parameters:
#   input file  = data_s04/lot_20210210_1130.fits
#   output file = 1130_2.png
#   resolution of output image = 450 dpi
#   minimum value to plot      = 550.000000
#   maximum value to plot      = 650.000000
#
# now, reading a FITS file...
# finished reading a FITS file!
# now, writing an image file...
# data_s04/lot_20210210_1130.fits ==> 1130_2.png
# finished writing an image file!
% ls -lF 1130*.png
-rw-r--r--  1 daisuke  taiwan   182545 Mar 10 22:55 1130.png
-rw-r--r--  1 daisuke  taiwan  3914810 Mar 10 23:11 1130_2.png
```

Show the image. (Fig. 11) Now, it looks better.

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

Add a function to choose a colour map.

Python Code 14: advobs202202_s04_14.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing astropy module
```

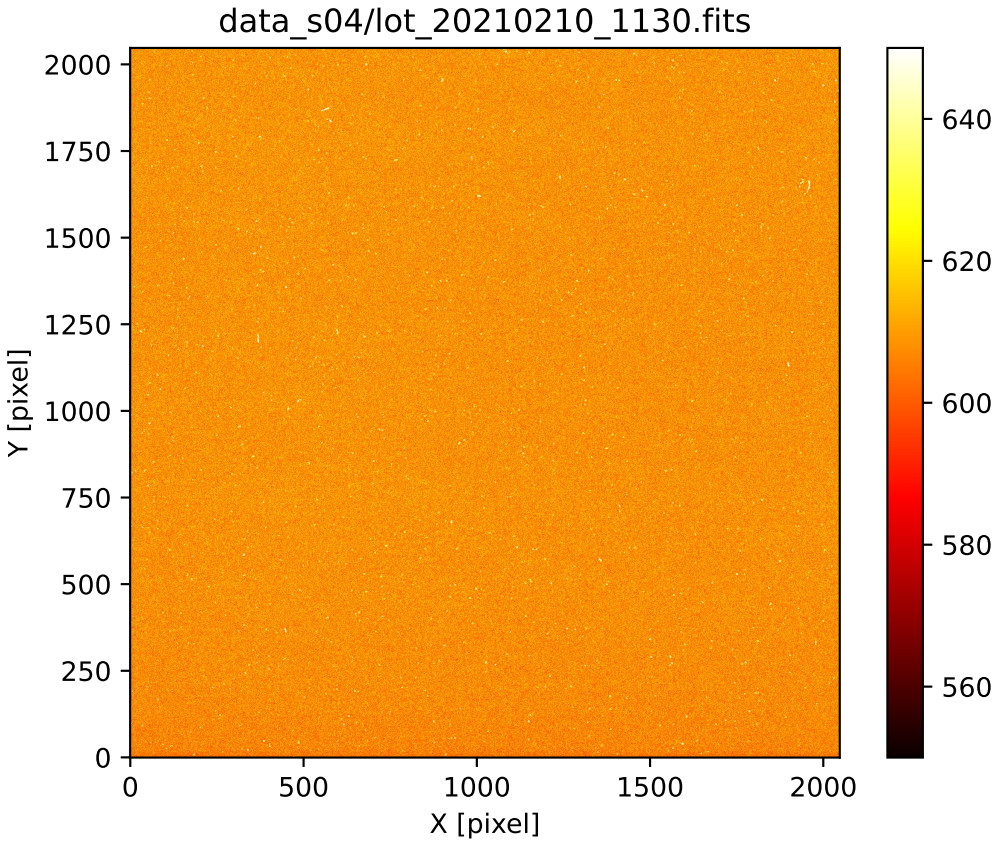

Figure 11: The image of the file `lot_20210210_1130.fits` using the range [550, 650].

```
import astropy.io.fits

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

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

# 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 ('-c', '--cmap', default='hot', choices=choices_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')
parser.add_argument ('-r', '--resolution', type=int, default=450, \
                     help='resolution of output file (default: 450 dpi)')

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

# input file existence check
path_input = pathlib.Path (file_input)
```

```
if not (path_input.exists ()):
    # printing a message
    print ("Error: the file \"%s\" does not exists!" % file_input)
    # exit
    sys.exit ()

# output file existence check
path_output = pathlib.Path (file_output)
if (path_output.exists ()):
    # printing a message
    print ("Error: the file \"%s\" exists!" % file_output)
    # exit
    sys.exit ()

# printing input parameters
print ("#")
print ("# Input parameters:")
print ("#   input file  = %s" % file_input)
print ("#   output file = %s" % file_output)
print ("#   resolution of output image = %d dpi" % resolution)
print ("#   minimum value to plot      = %f" % vmin)
print ("#   maximum value to plot      = %f" % vmax)
print ("#")

# printing status
print ("# now, reading a FITS file...")

# 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 ()

# printing status
print ("# finished reading a FITS file!")

# printing status
print ("# now, writing an image file...")

# 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=resolution)

# printing status
print ("# finished writing an image file!")
```

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

```
% chmod a+x advobs202202_s04_14.py
% ./advobs202202_s04_14.py -h
usage: advobs202202_s04_14.py [-h] [-a MIN] [-b MAX]
                              [-c {viridis,plasma,inferno,magma,cividis,binary,g
ray,bone,pink,spring,summer,autumn,winter,cool,hot,copper,ocean,terrain,gnuplot,
cubehelix,jet,turbo}]
                              [-i INPUT] [-o OUTPUT] [-r RESOLUTION]

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,binary,gray,bone,pink,spring,summer,a
utumn,winter,cool,hot,copper,ocean,terrain,gnuplot,cubehelix,jet,turbo}, --cmap
{viridis,plasma,inferno,magma,cividis,binary,gray,bone,pink,spring,summer,autumn
,winter,cool,hot,copper,ocean,terrain,gnuplot,cubehelix,jet,turbo}
                        maximum pixel value
  -i INPUT, --input INPUT
                        intput FITS file
  -o OUTPUT, --output OUTPUT
                        output image file
  -r RESOLUTION, --resolution RESOLUTION
                        resolution of output file (default: 450 dpi)

% ./advobs202202_s04_14.py -i data_s04/lot_20210210_1130.fits -o 1130_3.png \
? -a 550 -b 650 -c magma
#
# Input parameters:
#   input file  = data_s04/lot_20210210_1130.fits
#   output file = 1130_3.png
#   resolution of output image = 450 dpi
#   minimum value to plot      = 550.000000
#   maximum value to plot      = 650.000000
#
# now, reading a FITS file...
# finished reading a FITS file!
# now, writing an image file...
# data_s04/lot_20210210_1130.fits ==> 1130_3.png
# finished writing an image file!
% ls -lF 1130*.png
-rw-r--r--  1 daisuke  taiwan   182545 Mar 10 22:55 1130.png
-rw-r--r--  1 daisuke  taiwan  3914810 Mar 10 23:11 1130_2.png
-rw-r--r--  1 daisuke  taiwan  7101037 Mar 10 23:33 1130_3.png
```

Display the image. (Fig. 12)

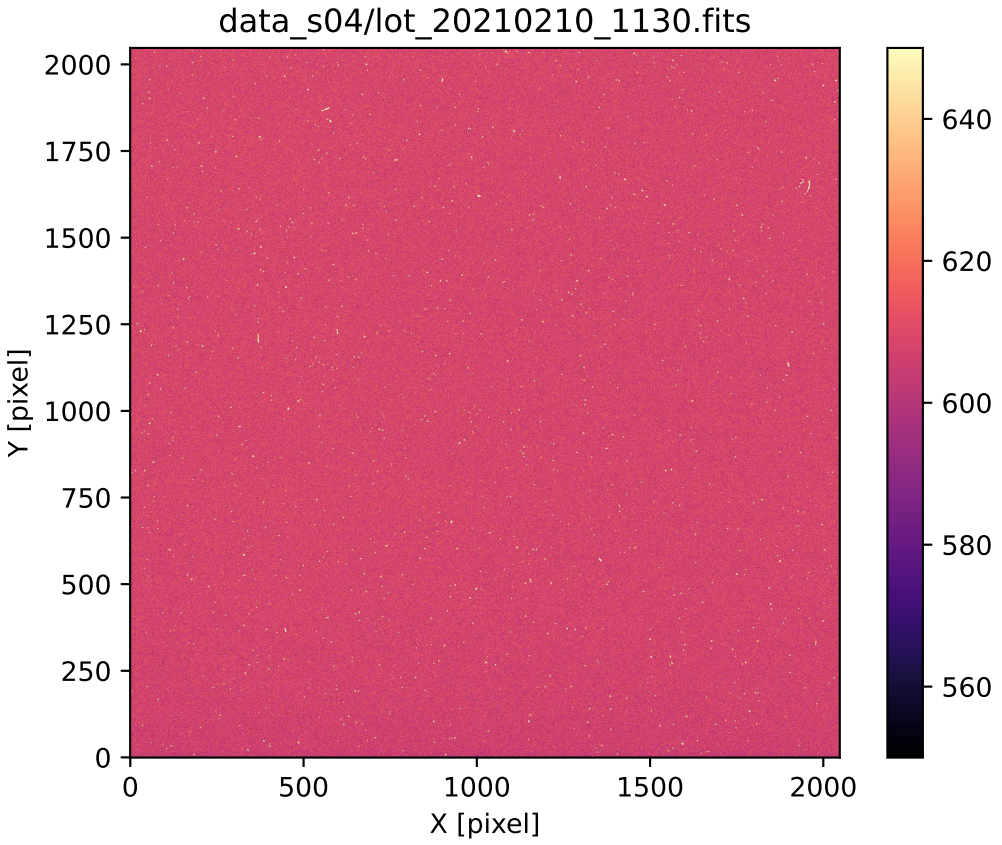

Figure 12: The image of the file `lot_20210210_1130.fits` using the range [550, 650] with colour map "magma".

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 15: advobs202202_s04_15.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing datetime module
import datetime

# importing numpy module
import numpy

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

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

# adding arguments
choices_rejection = ['none', 'sigclip']
choices_cenfunc   = ['mean', 'median']
parser.add_argument ('-r', '--rejection', choices=choices_rejection, \
                     default='none', \
                     help='outlier rejection algorithm (default: none)')
parser.add_argument ('-t', '--threshold', type=float, default=4.0, \
                     help='rejection threshold in sigma (default: 4.0)')
parser.add_argument ('-n', '--maxiters', type=int, default=10, \
                     help='maximum number of iterations')
parser.add_argument ('-c', '--cenfunc', choices=choices_cenfunc, \
                     default='mean', \
                     help='method to estimate centre value (default: mean)')
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
list_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 (list_input) < 2 ):
    # if the number of input files is less than 2, then stop the script
    print ("ERROR: Number of input files must be 2 or more!")
    # exit the script
    sys.exit ()

# checking input files
for file_fits in list_input:
    # if the file is not a FITS file, then stop the script
    if not (file_fits[-5:] == '.fits'):
        # printing error message
        print ("ERROR: Input files must be FITS files!")
        print ("ERROR: The file \"%s\" is not a FITS file!" % file_fits)
        # exit the script
        sys.exit ()
    # existence check
    path_fits = pathlib.Path (file_fits)
    if not (path_fits.exists ()):
        print ("ERROR: file \"%s\" does not exist!")
        # 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 ("ERROR: Output file must be FITS files!")
    # exit the script
    sys.exit ()
# existence check of output file
path_output = pathlib.Path (file_output)
if (path_output.exists ()):
    # printing error message
    print ("ERROR: output file \"%s\" exists!" % file_output)
    # exit the script
    sys.exit ()

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

# printing input parameters
print ("#")
print ("# Input parameters:")
print ("#   input FITS files:")
for file_fits in list_input:
    print ("#     %s" % file_fits)
print ("#   output FITS file = %s" % file_fits)
print ("#   rejection method = %s" % rejection)
print ("#   threshold        = %f sigma" % threshold)
print ("#   cenfunc          = %s" % cenfunc)
print ("#   maxiters         = %d" % maxiters)
print ("#")

# reading FITS files and constructing a data cube
```

```
for i in range (len (list_input)):
    # file name
    file_fits = list_input[i]

    # printing status
    print ("# now, reading FITS file \"%s\"..." % file_fits)

    # 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 )

    # printing status
    print ("# finished reading FITS file \"%s\"!" % file_fits)

# printing status
print ("# now, combining FITS files...")

# 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)

# printing status
print ("# finished combining FITS files!")

# printing status
print ("# now, writing FITS file \"%s\"..." % file_output)

# 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 list_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)

# printing status
print ("# finished writing FITS file \"%s\"!" % file_output)
```

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

```
% chmod a+x advobs202202_s04_15.py
% ./advobs202202_s04_15.py -h
usage: advobs202202_s04_15.py [-h] [-r {none,sigclip}] [-t THRESHOLD]
                              [-n MAXITERS] [-c {mean,median}] [-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 (default: none)
  -t THRESHOLD, --threshold THRESHOLD
                        rejection threshold in sigma (default: 4.0)
  -n MAXITERS, --maxiters MAXITERS
                        maximum number of iterations
  -c {mean,median}, --cenfunc {mean,median}
                        method to estimate centre value (default: mean)
  -o OUTPUT, --output OUTPUT
                        output FITS file

% ./advobs202202_s04_15.py -o dark3600_simple.fits \
? data_s04/lot_20210210_113?.fits
#
# Input parameters:
#   input FITS files:
#     data_s04/lot_20210210_1130.fits
#     data_s04/lot_20210210_1131.fits
#     data_s04/lot_20210210_1132.fits
#     data_s04/lot_20210210_1133.fits
#     data_s04/lot_20210210_1134.fits
#     data_s04/lot_20210210_1135.fits
#     data_s04/lot_20210210_1136.fits
#     data_s04/lot_20210210_1137.fits
#     data_s04/lot_20210210_1138.fits
#     data_s04/lot_20210210_1139.fits
#   output FITS file = data_s04/lot_20210210_1139.fits
#   rejection method = none
#   threshold        = 4.000000 sigma
```

```
#   cenfunc          = mean
#   maxiters         = 10
#
# now, reading FITS file "data_s04/lot_20210210_1130.fits"...
# finished reading FITS file "data_s04/lot_20210210_1130.fits"!
# now, reading FITS file "data_s04/lot_20210210_1131.fits"...
# finished reading FITS file "data_s04/lot_20210210_1131.fits"!
# now, reading FITS file "data_s04/lot_20210210_1132.fits"...
# finished reading FITS file "data_s04/lot_20210210_1132.fits"!
# now, reading FITS file "data_s04/lot_20210210_1133.fits"...
# finished reading FITS file "data_s04/lot_20210210_1133.fits"!
# now, reading FITS file "data_s04/lot_20210210_1134.fits"...
# finished reading FITS file "data_s04/lot_20210210_1134.fits"!
# now, reading FITS file "data_s04/lot_20210210_1135.fits"...
# finished reading FITS file "data_s04/lot_20210210_1135.fits"!
# now, reading FITS file "data_s04/lot_20210210_1136.fits"...
# finished reading FITS file "data_s04/lot_20210210_1136.fits"!
# now, reading FITS file "data_s04/lot_20210210_1137.fits"...
# finished reading FITS file "data_s04/lot_20210210_1137.fits"!
# now, reading FITS file "data_s04/lot_20210210_1138.fits"...
# finished reading FITS file "data_s04/lot_20210210_1138.fits"!
# now, reading FITS file "data_s04/lot_20210210_1139.fits"...
# finished reading FITS file "data_s04/lot_20210210_1139.fits"!
# now, combining FITS files...
# finished combining FITS files!
# now, writing FITS file "dark3600_simple.fits"...
# finished writing FITS file "dark3600_simple.fits"!
% ls -lF dark3600_simple.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 10 23:58 dark3600_simple.fits
```

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

Python Code 16: advobs202202_s04_16.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing astropy module
import astropy.io.fits

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

# adding arguments
parser.add_argument ('file', default='', 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
if not (file_input[-5:] == '.fits'):
    # printing error message
    print ("Input file must be FITS files!")
    # exit the script
    sys.exit ()
# if the file does not exist, then stop the script
path_input = pathlib.Path (file_input)
if not (path_input.exists ()):
    # printing error message
    print ("Input file FITS file does not exist!")
    # exit the script
    sys.exit ()

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

# 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 the FITS file `dark3600_simple.fits`.

```
% chmod a+x advobs202202_s04_16.py
% ./advobs202202_s04_16.py -h
usage: advobs202202_s04_16.py [-h] file

Printing FITS header

positional arguments:
  file        input FITS file

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

% ./advobs202202_s04_16.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
DATE-OBS= '2021-02-10' /         YYYY-MM-DDThh:mm:ss observation start, UT
TIME-OBS= '23:58:08' /           HH:MM:SS observation start time, UT
EXPTIME =   3600.0000000000000 /Exposure time in seconds
EXPOSURE=   3600.0000000000000 /Exposure time in seconds
SET-TEMP=  -80.000000000000000 /CCD temperature setpoint in C
CCD-TEMP=  -80.000000000000000 /CCD temperature at start of exposure in C
```

```
XPIXSZ  =   15.000000000000000 /Pixel Width in microns (after binning)
YPIXSZ  =   15.000000000000000 /Pixel Height in microns (after binning)
XBINNING=                    1 /Binning factor in width
YBINNING=                    1 /Binning factor in height
XORGSUBF=                    0 /Subframe X position in binned pixels
YORGSUBF=                    0 /Subframe Y position in binned pixels
IMAGETYP= 'DARK    ' /           Type of image
OBJCTRA = '18 25 53' /           Nominal Right Ascension of center of image
OBJCTDEC= '+25 27 15' /          Nominal Declination of center of image
OBJCTALT= ' 88.0004' /           Nominal altitude of center of image
OBJCTAZ = '358.4651' /           Nominal azimuth of center of image
OBJCTHA = '  0.0040' /           Nominal hour angle of center of image
SITELAT = '23 28 07' /           Latitude of the imaging location
SITELONG= '120 52 25' /          Longitude of the imaging location
JD      =   2459256.4987037037 /Julian Date at start of exposure
JD-HELIO=   2459256.5168002900 /Heliocentric Julian Date at exposure midpoint
AIRMASS =   1.0295841535794146 /Relative optical path length through atmosphere
FOCALLEN=   8000.0000000000000 /Focal length of telescope in mm
APTDIA  =   1000.0000000000000 /Aperture diameter of telescope in mm
APTAREA =   772124.95592236519 /Aperture area of telescope in mm^2
SWCREATE= 'MaxIm DL Version 5.24 130419 0CYVP' /Name of software that created
         the image
OBJECT  = 'dark    '
TELESCOP= 'LOT     ' /           telescope used to acquire this image
INSTRUME= 'Driver for Princeton Instruments cameras'
OBSERVER= 'Kinoshita Daisuke'
NOTES   = '        '
DETECTOR= '        '
OWNER   = 'Institute of Astronomy, NCU, Taiwan'
TIMESYS = 'UTC     '
EQUINX  = '2000    '
EPOCH   = '2000    '
RADECSYS= 'FK5     '
CAMERA  = 'SOPHIA  '
GAIN    = '2       '
SITEELEV=                 2862
RMSNOISE= '8.5     '
OPERATOR= 'lulin   '
CTYPE1  = 'RA---TAN' /           fastest changing axis name
CTYPE2  = 'DEC--TAN' /           slowest changing axis name
FOV     = '13'' 08" x 13'' 08"'
PIXSIZE =  0.39000000000000001
FLIPSTAT= '        '
SWOWNER = 'Ming-Hsin Chang' /   Licensed owner of software
HISTORY FITS file created by the command "./advobs202202_s04_15.py"
HISTORY Updated on 2022-03-10T23:58:50.100603
COMMENT List of combined files:
COMMENT   data_s04/lot_20210210_1130.fits
COMMENT   data_s04/lot_20210210_1131.fits
COMMENT   data_s04/lot_20210210_1132.fits
COMMENT   data_s04/lot_20210210_1133.fits
COMMENT   data_s04/lot_20210210_1134.fits
COMMENT   data_s04/lot_20210210_1135.fits
COMMENT   data_s04/lot_20210210_1136.fits
COMMENT   data_s04/lot_20210210_1137.fits
COMMENT   data_s04/lot_20210210_1138.fits
COMMENT   data_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.

```
% ./advobs202202_s04_15.py -o dark3600_sigclip.fits -r sigclip -t 3 \
? -c median data_s04/lot_20210210_113?.fits
#
# Input parameters:
#   input FITS files:
#     data_s04/lot_20210210_1130.fits
#     data_s04/lot_20210210_1131.fits
#     data_s04/lot_20210210_1132.fits
#     data_s04/lot_20210210_1133.fits
#     data_s04/lot_20210210_1134.fits
#     data_s04/lot_20210210_1135.fits
#     data_s04/lot_20210210_1136.fits
#     data_s04/lot_20210210_1137.fits
#     data_s04/lot_20210210_1138.fits
#     data_s04/lot_20210210_1139.fits
#   output FITS file = data_s04/lot_20210210_1139.fits
#   rejection method = sigclip
#   threshold        = 3.000000 sigma
#   cenfunc          = median
#   maxiters         = 10
#
# now, reading FITS file "data_s04/lot_20210210_1130.fits"...
# finished reading FITS file "data_s04/lot_20210210_1130.fits"!
# now, reading FITS file "data_s04/lot_20210210_1131.fits"...
# finished reading FITS file "data_s04/lot_20210210_1131.fits"!
# now, reading FITS file "data_s04/lot_20210210_1132.fits"...
# finished reading FITS file "data_s04/lot_20210210_1132.fits"!
# now, reading FITS file "data_s04/lot_20210210_1133.fits"...
# finished reading FITS file "data_s04/lot_20210210_1133.fits"!
# now, reading FITS file "data_s04/lot_20210210_1134.fits"...
# finished reading FITS file "data_s04/lot_20210210_1134.fits"!
# now, reading FITS file "data_s04/lot_20210210_1135.fits"...
# finished reading FITS file "data_s04/lot_20210210_1135.fits"!
# now, reading FITS file "data_s04/lot_20210210_1136.fits"...
# finished reading FITS file "data_s04/lot_20210210_1136.fits"!
# now, reading FITS file "data_s04/lot_20210210_1137.fits"...
# finished reading FITS file "data_s04/lot_20210210_1137.fits"!
# now, reading FITS file "data_s04/lot_20210210_1138.fits"...
# finished reading FITS file "data_s04/lot_20210210_1138.fits"!
# now, reading FITS file "data_s04/lot_20210210_1139.fits"...
# finished reading FITS file "data_s04/lot_20210210_1139.fits"!
# now, combining FITS files...
# finished combining FITS files!
# now, writing FITS file "dark3600_sigclip.fits"...
# finished writing FITS file "dark3600_sigclip.fits"!
% ls -lF dark3600_*.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 11 00:11 dark3600_sigclip.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 10 23:58 dark3600_simple.fits
```

Print the header of the file `dark3600_sigclip.fits`.

```
% ./advobs202202_s04_16.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
DATE-OBS= '2021-02-10' /        YYYY-MM-DDThh:mm:ss observation start, UT
TIME-OBS= '23:58:08' /          HH:MM:SS observation start time, UT
EXPTIME =   3600.0000000000000 /Exposure time in seconds
EXPOSURE=   3600.0000000000000 /Exposure time in seconds
SET-TEMP=  -80.000000000000000 /CCD temperature setpoint in C

.....

HISTORY FITS file created by the command "./advobs202202_s04_15.py"
HISTORY Updated on 2022-03-11T00:11:11.877986
COMMENT List of combined files:
COMMENT   data_s04/lot_20210210_1130.fits
COMMENT   data_s04/lot_20210210_1131.fits
COMMENT   data_s04/lot_20210210_1132.fits
COMMENT   data_s04/lot_20210210_1133.fits
COMMENT   data_s04/lot_20210210_1134.fits
COMMENT   data_s04/lot_20210210_1135.fits
COMMENT   data_s04/lot_20210210_1136.fits
COMMENT   data_s04/lot_20210210_1137.fits
COMMENT   data_s04/lot_20210210_1138.fits
COMMENT   data_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.

```
% ./advobs202202_s04_10.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
-------------------------------------------------------------------------------
```

Read two FITS files and generate PNG files.

```
% ./advobs202202_s04_14.py -a 590 -b 650 -c viridis \
? -i dark3600_simple.fits -o dark3600_simple.png
#
# Input parameters:
#   input file  = dark3600_simple.fits
#   output file = dark3600_simple.png
#   resolution of output image = 450 dpi
#   minimum value to plot      = 590.000000
#   maximum value to plot      = 650.000000
#
# now, reading a FITS file...
```

```
# finished reading a FITS file!
# now, writing an image file...
# dark3600_simple.fits ==> dark3600_simple.png
# finished writing an image file!
% ./advobs202202_s04_14.py -a 590 -b 650 -c viridis \
? -i dark3600_sigclip.fits -o dark3600_sigclip.png
#
# Input parameters:
#   input file  = dark3600_sigclip.fits
#   output file = dark3600_sigclip.png
#   resolution of output image = 450 dpi
#   minimum value to plot      = 590.000000
#   maximum value to plot      = 650.000000
#
# now, reading a FITS file...
# finished reading a FITS file!
# now, writing an image file...
# dark3600_sigclip.fits ==> dark3600_sigclip.png
# finished writing an image file!
% ls -lF dark3600_si*
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 11 00:11 dark3600_sigclip.fits
-rw-r--r--  1 daisuke  taiwan   4216934 Mar 11 00:18 dark3600_sigclip.png
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 10 23:58 dark3600_simple.fits
-rw-r--r--  1 daisuke  taiwan   4424142 Mar 11 00:17 dark3600_simple.png
```

Show the image of `dark3600_simple.fits` (Fig. 13) and `dark3600_sigclip.fits` (Fig. 14). Many cosmic rays are removed by sigma clipping.

```
% feh -dF dark3600_*.png
```

8 Estimating dark current generation rate

Estimate dark current generation rate.

8.1 Combining dark frames

Combine dark frames of 10-sec exposure.

```
% ./advobs202202_s04_15.py -r sigclip -t 3 -c median -o d0010.fits */*_103?.fits
#
# Input parameters:
#   input FITS files:
#     data_s04/lot_20210210_1030.fits
#     data_s04/lot_20210210_1031.fits
#     data_s04/lot_20210210_1032.fits
#     data_s04/lot_20210210_1033.fits
#     data_s04/lot_20210210_1034.fits
#     data_s04/lot_20210210_1035.fits
#     data_s04/lot_20210210_1036.fits
#     data_s04/lot_20210210_1037.fits
#     data_s04/lot_20210210_1038.fits
#     data_s04/lot_20210210_1039.fits
#   output FITS file = data_s04/lot_20210210_1039.fits
#   rejection method = sigclip
#   threshold        = 3.000000 sigma
#   cenfunc          = median
```

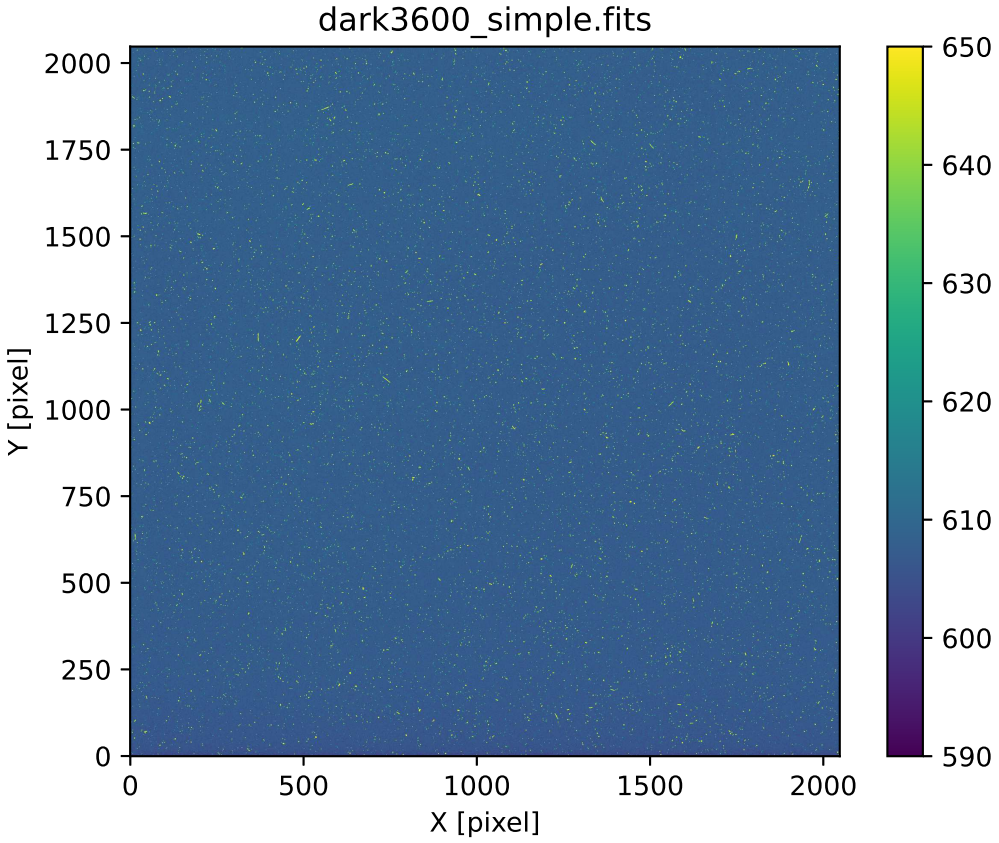

Figure 13: Combined 3600-sec dark frame using simple mean.

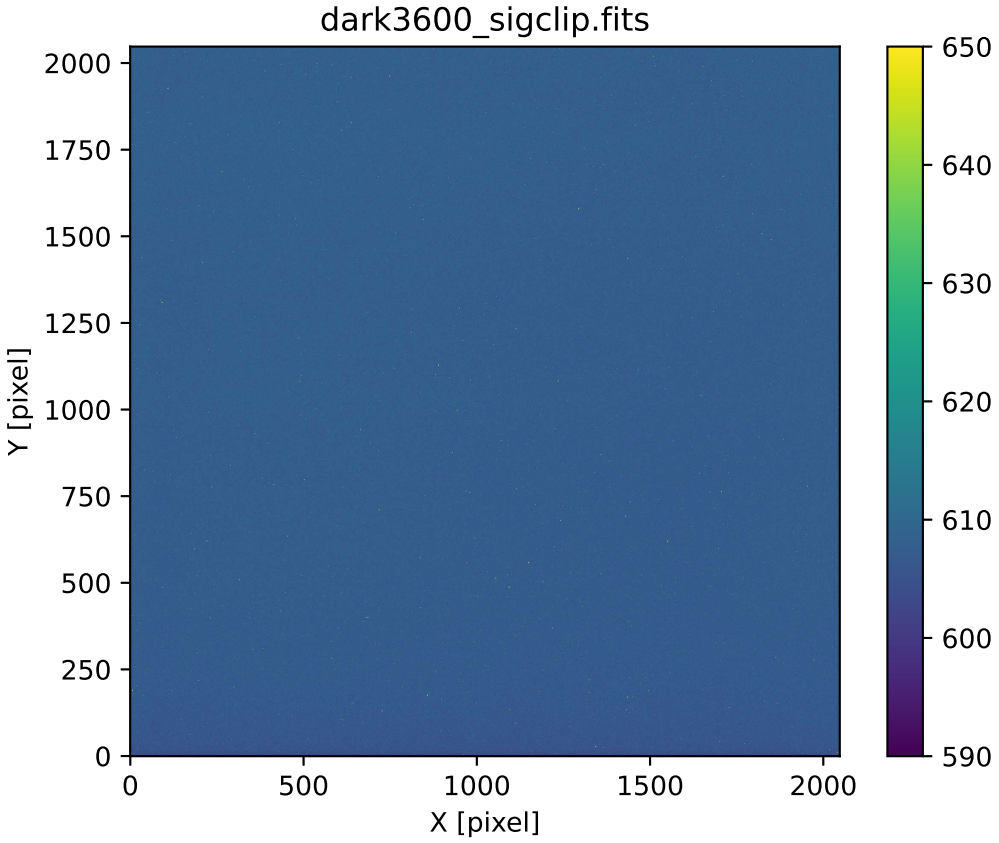

Figure 14: Combined 3600-sec dark frame using sigma clipping.

```
#   maxiters         = 10
#
# now, reading FITS file "data_s04/lot_20210210_1030.fits"...
# finished reading FITS file "data_s04/lot_20210210_1030.fits"!
# now, reading FITS file "data_s04/lot_20210210_1031.fits"...
# finished reading FITS file "data_s04/lot_20210210_1031.fits"!
# now, reading FITS file "data_s04/lot_20210210_1032.fits"...
# finished reading FITS file "data_s04/lot_20210210_1032.fits"!
# now, reading FITS file "data_s04/lot_20210210_1033.fits"...
# finished reading FITS file "data_s04/lot_20210210_1033.fits"!
# now, reading FITS file "data_s04/lot_20210210_1034.fits"...
# finished reading FITS file "data_s04/lot_20210210_1034.fits"!
# now, reading FITS file "data_s04/lot_20210210_1035.fits"...
# finished reading FITS file "data_s04/lot_20210210_1035.fits"!
# now, reading FITS file "data_s04/lot_20210210_1036.fits"...
# finished reading FITS file "data_s04/lot_20210210_1036.fits"!
# now, reading FITS file "data_s04/lot_20210210_1037.fits"...
# finished reading FITS file "data_s04/lot_20210210_1037.fits"!
# now, reading FITS file "data_s04/lot_20210210_1038.fits"...
# finished reading FITS file "data_s04/lot_20210210_1038.fits"!
# now, reading FITS file "data_s04/lot_20210210_1039.fits"...
# finished reading FITS file "data_s04/lot_20210210_1039.fits"!
# now, combining FITS files...
# finished combining FITS files!
# now, writing FITS file "d0010.fits"...
# finished writing FITS file "d0010.fits"!
% ls -lF d0010.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 11 00:33 d0010.fits
```

Similarly, make combined dark frames of 30-sec, 90-sec, 300-sec, 900-sec, and 3600-sec.

```
% ./advobs202202_s04_15.py -r sigclip -t 3 -c median -o d0030.fits */*_105?.fits
% ./advobs202202_s04_15.py -r sigclip -t 3 -c median -o d0090.fits */*_107?.fits
% ./advobs202202_s04_15.py -r sigclip -t 3 -c median -o d0300.fits */*_109?.fits
% ./advobs202202_s04_15.py -r sigclip -t 3 -c median -o d0900.fits */*_111?.fits
% ./advobs202202_s04_15.py -r sigclip -t 3 -c median -o d3600.fits */*_113?.fits
% ls -lF d????.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 11 00:33 d0010.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Mar 11 00:34 d0030.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 11 00:35 d0090.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 11 00:35 d0300.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 11 00:36 d0900.fits
-rw-r--r--  1 daisuke  taiwan  33563520 Mar 11 00:36 d3600.fits
```

8.2 Calculating mean values of combined dark frames

Calculate mean values of combined dark frames.

```
% ./advobs202202_s04_10.py -r sigclip d????.fits
--------------------------------------------------------------------------------
file name                    n_pix     mean   median   stddev      min      max
================================================================================
d0010.fits                 4193636   606.44   606.50     2.60   596.10   616.80
d0030.fits                 4193583   606.54   606.50     2.58   596.22   616.80
d0090.fits                 4193563   606.53   606.50     2.60   596.20   616.90
d0300.fits                 4193456   606.42   606.40     2.60   596.10   616.80
d0900.fits                 4192607   606.62   606.60     2.59   596.30   616.90
```

```
d3600.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 17: advobs202202_s04_17.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing numpy module
import numpy

# importing astropy module
import astropy.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
choices_rejection = ['none', 'sigclip']
choices_cenfunc   = ['mean', 'median']
parser.add_argument ('-r', '--rejection', choices=choices_rejection, \
                     default='none', \
                     help='outlier rejection algorithm (default: none)')
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=choices_cenfunc, \
                     default='mean', \
                     help='method to estimate centre value (default: mean)')
parser.add_argument ('-o', '--output', default='dark.png', \
                     help='output image file')
parser.add_argument ('-d', '--resolution', type=int, default=450, \
                     help='resolution of output file (default: 450 dpi)')
parser.add_argument ('files', nargs='+', help='input FITS files')
```

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

# parameters given by command-line arguments
list_input  = args.files
file_output = args.output
rejection   = args.rejection
threshold   = args.threshold
cenfunc     = args.cenfunc
maxiters    = args.maxiters
resolution  = args.resolution

# checking input files
for file_fits in list_input:
    # if the file is not a FITS file, then stop the script
    if not (file_fits[-5:] == '.fits'):
        # printing error message
        print ("ERROR: Input files must be FITS files!")
        print ("ERROR: The file \"%s\" is not a FITS file!" % file_fits)
        # exit the script
        sys.exit ()
    # existence check
    path_fits = pathlib.Path (file_fits)
    if not (path_fits.exists ()):
        # printing error message
        print ("ERROR: the file \"%s\" does not exist!" % 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:] == '.eps') or (file_output[-4:] == '.pdf') \
         or (file_output[-4:] == '.png') or (file_output[-3:] == '.ps') ):
    # printing error message
    print ("Output file must be EPS, or PDF, or PNG, or PS file!")
    # exit the script
    sys.exit ()
# existence check
path_output = pathlib.Path (file_output)
if (path_output.exists ()):
    # printing error message
    print ("ERROR: the file \"%s\" exists!" % file_output)
    # exit the script
    sys.exit ()

# reading FITS files and calculating mean and stddev
for file_fits in list_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'):
        # calculation of mean, median, and stddev using sigma-clipping
        mean, median, stddev \
            = astropy.stats.sigma_clipped_stats (data0, sigma=threshold, \
                                                 maxiters=maxiters, \
                                                 cenfunc=cenfunc, \
                                                 stdfunc='std')
    elif (rejection == 'none'):
        # calculation of simple mean and stddev
        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 ('Dark Current Generation Rate')
ax.set_xlabel ('Exposure Time [sec]')
ax.set_ylabel ('Mean Pixel Value [ADU]')

# plotting a figure
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=resolution)
```

Run the script, and generate a plot.

```
% chmod a+x advobs202202_s04_17.py
% ./advobs202202_s04_17.py -h
usage: advobs202202_s04_17.py [-h] [-r {none,sigclip}] [-t THRESHOLD]
                              [-n MAXITERS] [-c {mean,median}] [-o OUTPUT]
```

```
                              [-d RESOLUTION]
                              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 (default: none)
  -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 (default: mean)
  -o OUTPUT, --output OUTPUT
                        output image file
  -d RESOLUTION, --resolution RESOLUTION
                        resolution of output file (default: 450 dpi)

% ./advobs202202_s04_17.py -r sigclip -o darkrate.png d????.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 -lF darkrate.png
-rw-r--r--  1 daisuke  taiwan  143562 Mar 11 00:53 darkrate.png
```

Show the plot. (Fig. 15)

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

8.4 Fitting using least squares method

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

Python Code 18: advobs202202_s04_18.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing numpy module
import numpy
```

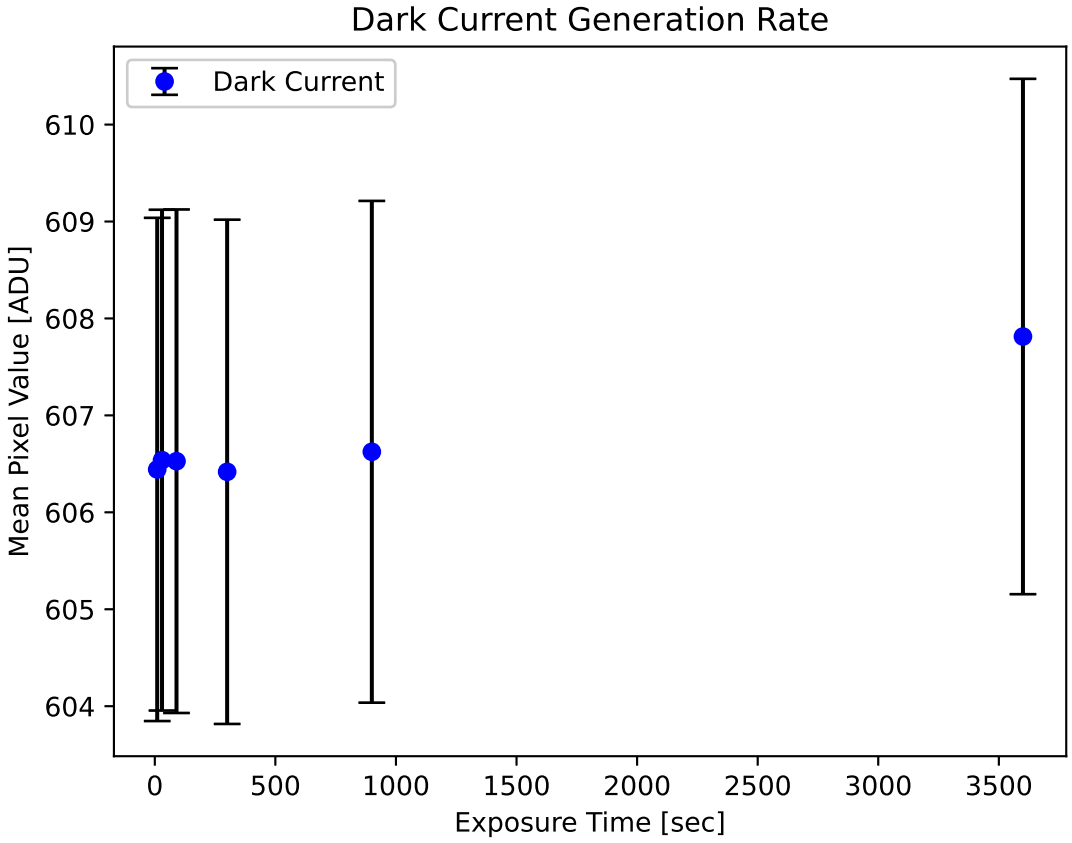

Figure 15: The plot of exposure time vs. mean pixel value of dark frame.

```
# 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
choices_rejection = ['none', 'sigclip']
choices_cenfunc   = ['mean', 'median']
parser.add_argument ('-r', '--rejection', choices=choices_rejection, \
                     default='none', \
                     help='outlier rejection algorithm (default: none)')
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=choices_cenfunc, \
                     default='mean', \
                     help='method to estimate centre value (default: mean)')
parser.add_argument ('-o', '--output', default='dark.png', \
                     help='output image file')
parser.add_argument ('-d', '--resolution', type=int, default=450, \
                     help='resolution of output file (default: 450 dpi)')
parser.add_argument ('files', nargs='+', help='input FITS files')

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

# parameters given by command-line arguments
list_input  = args.files
file_output = args.output
rejection   = args.rejection
threshold   = args.threshold
cenfunc     = args.cenfunc
maxiters    = args.maxiters
resolution  = args.resolution

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

```
        # exit the script
        sys.exit ()
    # existence check
    path_fits = pathlib.Path (file_fits)
    if not (path_fits.exists ()):
        # printing error message
        print ("ERROR: the file \"%s\" does not exist!" % 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:] == '.eps') or (file_output[-4:] == '.pdf') \
         or (file_output[-4:] == '.png') or (file_output[-3:] == '.ps') ):
    # printing error message
    print ("Output file must be EPS, or PDF, or PNG, or PS file!")
    # exit the script
    sys.exit ()
# existence check
path_output = pathlib.Path (file_output)
if (path_output.exists ()):
    # printing error message
    print ("ERROR: the file \"%s\" exists!" % file_output)
    # exit the script
    sys.exit ()

# reading FITS files and calculating mean and stddev
for file_fits in list_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'):
        # calculation of mean, median, and stddev using sigma-clipping
        mean, median, stddev \
            = astropy.stats.sigma_clipped_stats (data0, sigma=threshold, \
                                                 maxiters=maxiters, \
                                                 cenfunc=cenfunc, \
                                                 stdfunc='std')
    elif (rejection == 'none'):
        # calculation of simple mean and stddev
        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 using SciPy
#

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

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

# weighted least-squares fitting
# x: data_exptime
# y: data_mean
# Delta y: data_stddev
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 ('Dark Current Generation Rate')
ax.set_xlabel ('Exposure Time [sec]')
ax.set_ylabel ('Mean Pixel Value [ADU]')

# plotting a figure
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=resolution)
```

Execute the script, and carry out least squares fitting.

```
% chmod a+x advobs202202_s04_18.py
% ./advobs202202_s04_18.py -h
usage: advobs202202_s04_18.py [-h] [-r {none,sigclip}] [-t THRESHOLD]
                              [-n MAXITERS] [-c {mean,median}] [-o OUTPUT]
                              [-d RESOLUTION]
                              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 (default: none)
  -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 (default: mean)
  -o OUTPUT, --output OUTPUT
                        output image file
  -d RESOLUTION, --resolution RESOLUTION
                        resolution of output file (default: 450 dpi)
```

```
% ./advobs202202_s04_18.py -r sigclip -o darkrate_fit.png d????.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 -lF darkrate*.png
-rw-r--r--  1 daisuke  taiwan  143562 Mar 11 00:53 darkrate.png
-rw-r--r--  1 daisuke  taiwan  176576 Mar 11 01:01 darkrate_fit.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. 16)

```
% feh -dF darkrate_fit.png
```

9 For your further reading

- 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`

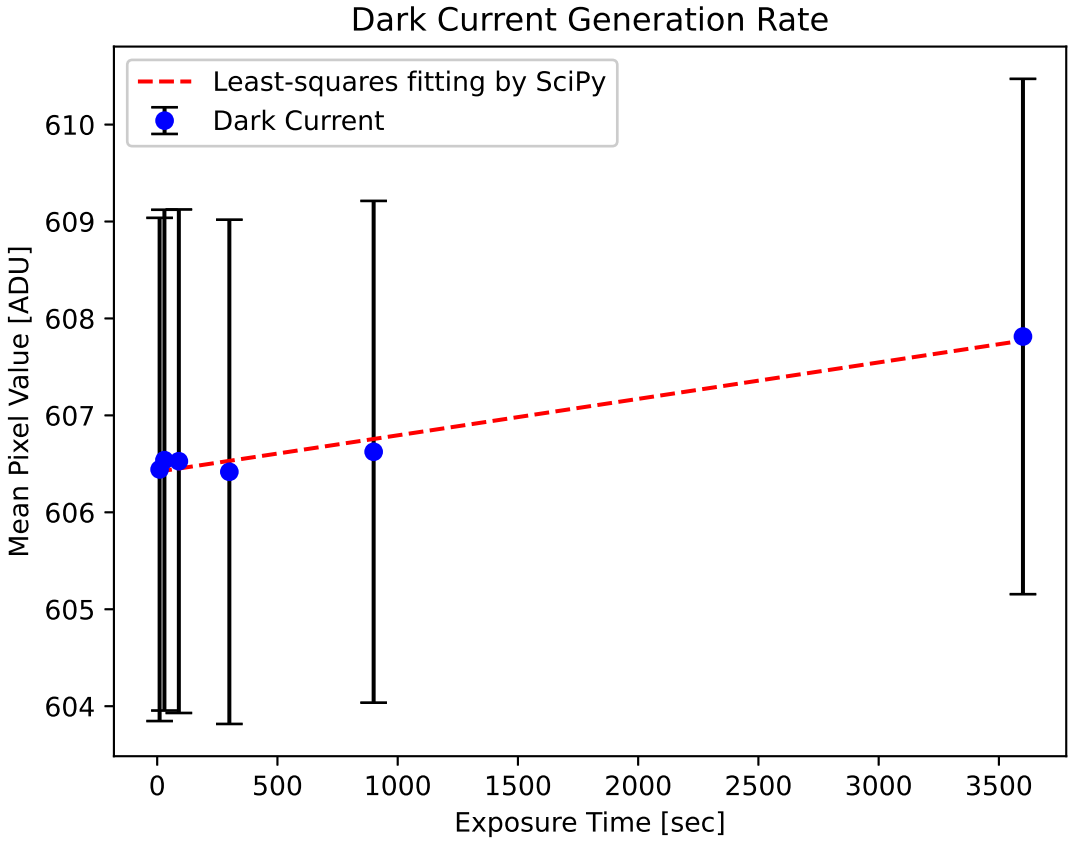

Figure 16: The result of least squares fitting.

10 Exercise

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.