Advanced Astronomical Observations 2021
Session 03: Examining Bias Frames

Kinoshita Daisuke

03 March 2021
publicly accessible version

About this file...

- Important information about this file
  - The author of this file is Kinoshita Daisuke.
  - The original version of this file was used for the course "Advanced Astronomical Observations" (course ID: AS6005) offered at Institute of Astronomy, National Central University from February 2021 to June 2021.
  - The file is provided in the hope that it will be useful, but there is no guarantee for the correctness. Use this file at your own risk.
  - If you are willing to use this file for your study, please feel free to use. I'll be very happy to receive feedback from you.
  - If you are willing to use this file for your teaching, please contact to Kinoshita Daisuke. When you use this file partly or entirely, please mention clearly that the author of the original version is Kinoshita Daisuke. Please help me to improve the contents of this file by sending your feedback.
  - Contact address: `https://www.instagram.com/daisuke23888/`

For this session, we examine bias frames. Bias frames are images of 0-sec exposure. Those data are taken when the shutter is closed.

1 Downloading data

A set of FITS files for this session is placed at following location. Download the file. The size of the file is about 770 MB.

- `https://s3b.astro.ncu.edu.tw/advobs_202102/data/data_ao2021_s03.tar.xz`

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

```
% curl -k -o data_ao2021_s03.tar.xz \
? https://s3b.astro.ncu.edu.tw/advobs_202102/data/data_ao2021_s03.tar.xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  772M  100  772M    0     0  1456k      0  0:09:03  0:09:03 --:--:-- 4210k
% ls -l data_ao2021_s03.tar.xz
-rw-r--r--  1 daisuke  taiwan  810521012 Mar  1 23:20 data_ao2021_s03.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. 285 FITS files should be extracted from the archive file.

```
% ls -l data_ao2021_s03.tar.xz
-rw-r--r--  1 daisuke  taiwan  810521012 Mar  1 23:20 data_ao2021_s03.tar.xz
% tar xJvf data_ao2021_s03.tar.xz
x data_ao2021_s03/
x data_ao2021_s03/lot_20210212_0079.fits
x data_ao2021_s03/lot_20210212_0080.fits
x data_ao2021_s03/lot_20210212_0081.fits
x data_ao2021_s03/lot_20210212_0082.fits
x data_ao2021_s03/lot_20210212_0083.fits

.....

x data_ao2021_s03/lot_20210212_0814.fits
x data_ao2021_s03/lot_20210212_0815.fits
x data_ao2021_s03/lot_20210212_0816.fits
x data_ao2021_s03/lot_20210212_0817.fits
x data_ao2021_s03/lot_20210212_0818.fits
% ls data_ao2021_s03/ | head
lot_20210212_0079.fits
lot_20210212_0080.fits
lot_20210212_0081.fits
lot_20210212_0082.fits
lot_20210212_0083.fits
lot_20210212_0084.fits
lot_20210212_0100.fits
lot_20210212_0101.fits
lot_20210212_0102.fits
lot_20210212_0112.fits
% ls data_ao2021_s03/*.fits | wc
     285     285   11115
```

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

```
% unxz -c data_ao2021_s03.tar.xz | tar xvf -
x data_ao2021_s03/
x data_ao2021_s03/lot_20210212_0079.fits
x data_ao2021_s03/lot_20210212_0080.fits
x data_ao2021_s03/lot_20210212_0081.fits
x data_ao2021_s03/lot_20210212_0082.fits
x data_ao2021_s03/lot_20210212_0083.fits

.....

x data_ao2021_s03/lot_20210212_0814.fits
x data_ao2021_s03/lot_20210212_0815.fits
x data_ao2021_s03/lot_20210212_0816.fits
x data_ao2021_s03/lot_20210212_0817.fits
x data_ao2021_s03/lot_20210212_0818.fits
```

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

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

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

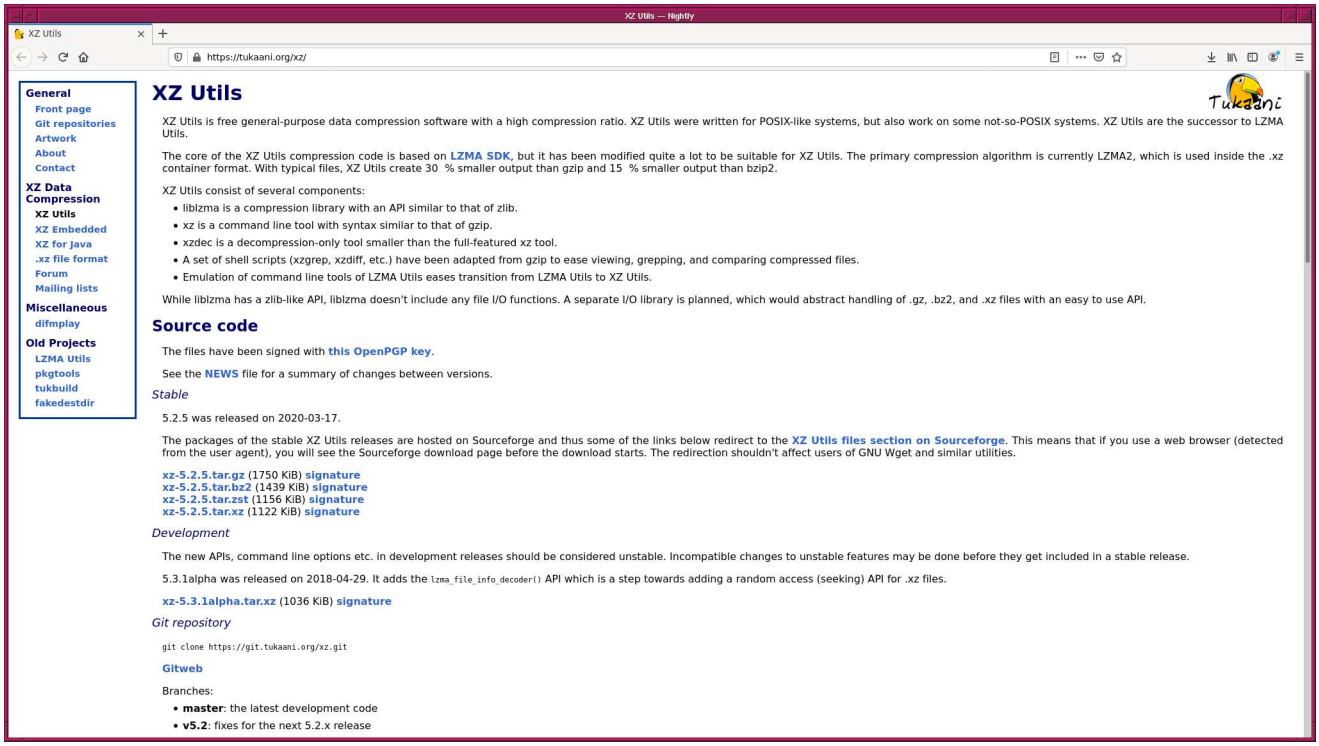

Figure 1: The website of XZ Utils.

```
% ls -l data_ao2021_s03.tar.xz
-rw-r--r--  1 daisuke  taiwan  810521012 Mar  1 23:20 data_ao2021_s03.tar.xz
% unxz data_ao2021_s03.tar.xz
% ls -l data_ao2021_s03.tar
-rw-r--r--  1 daisuke  taiwan  2392797696 Mar  1 23:20 data_ao2021_s03.tar
% tar xvf data_ao2021_s03.tar
x data_ao2021_s03/
x data_ao2021_s03/lot_20210212_0079.fits
x data_ao2021_s03/lot_20210212_0080.fits
x data_ao2021_s03/lot_20210212_0081.fits
x data_ao2021_s03/lot_20210212_0082.fits
x data_ao2021_s03/lot_20210212_0083.fits

.....

x data_ao2021_s03/lot_20210212_0814.fits
x data_ao2021_s03/lot_20210212_0815.fits
x data_ao2021_s03/lot_20210212_0816.fits
x data_ao2021_s03/lot_20210212_0817.fits
x data_ao2021_s03/lot_20210212_0818.fits
```

3 Checking the data type

Make a Python script to check the data type of FITS files.

Python Code 1: ao2021_s03_01.py

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

# importing argparse module
```

```
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Checking a keyword IMAGETYP'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('files', nargs='+', help='intput FITS files')

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

# input FITS file
list_file = args.files

# processing files
for file_fits in list_file:
    # if input file is not a FITS file, then skip
    if not (file_fits[-5:] == '.fits'):
        # printing a message
        print ("The file \"%s\" is not a FITS file!" % file_fits)
        # going to next file
        continue

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

    # primary HDU
    hdu0 = hdu_list[0]

    # reading header
    header0 = hdu0.header

    # closing FITS file
    hdu_list.close ()

    # printing a value of IMAGETYP keyword
    print ("%s : %s" % (file_fits, header0['IMAGETYP']) )
```

Execute the script, and check the data type of FITS files.

```
% ./ao2021_s03_01.py -h
usage: ao2021_s03_01.py [-h] files [files ...]

Checking a keyword IMAGETYP

positional arguments:
  files       intput FITS files

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

```
% ./ao2021_s03_01.py data_ao2021_s03/*.fits
data_ao2021_s03/lot_20210212_0079.fits : BIAS
data_ao2021_s03/lot_20210212_0080.fits : BIAS
data_ao2021_s03/lot_20210212_0081.fits : BIAS
data_ao2021_s03/lot_20210212_0082.fits : BIAS
data_ao2021_s03/lot_20210212_0083.fits : BIAS

.....

data_ao2021_s03/lot_20210212_0814.fits : BIAS
data_ao2021_s03/lot_20210212_0815.fits : BIAS
data_ao2021_s03/lot_20210212_0816.fits : BIAS
data_ao2021_s03/lot_20210212_0817.fits : BIAS
data_ao2021_s03/lot_20210212_0818.fits : BIAS
```

All the FITS files for this session are bias frames.

4 Reading a bias frame

Make a Python script to read a bias frame and show the shape of the data.

Python Code 2: ao2021_s03_02.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.io.fits

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

# adding arguments
parser.add_argument ('-i', '--input', default='test.fits', \
                     help='intput FITS file')

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

# input FITS file
file_input = args.input

# 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 ()

# 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 shape of data
print ("shape of data    =", data0.shape)
print ("number of pixels =", data0.size, "pixels")

# printing data
print ("pixel data:")
print (data0)
```

Choose one FITS file and execute the script.

```
% chmod a+x ao2021_s03_02.py
% ./ao2021_s03_02.py -h
usage: ao2021_s03_02.py [-h] [-i INPUT]

Reading a FITS file

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        intput FITS file

% ./ao2021_s03_02.py -i data_ao2021_s03/lot_20210212_0079.fits
shape of data    = (2048, 2048)
number of pixels = 4194304 pixels
pixel data:
[[602 612 621 ... 603 608 588]
 [592 597 598 ... 592 602 591]
 [573 600 598 ... 603 609 612]
 ...
 [600 586 595 ... 599 603 593]
 [598 604 602 ... 610 594 596]
 [591 601 603 ... 588 599 594]]
```

Choose the other FITS file, and try again.

```
% ./ao2021_s03_02.py -i data_ao2021_s03/lot_20210212_0080.fits
shape of data    = (2048, 2048)
number of pixels = 4194304 pixels
pixel data:
[[610 606 595 ... 604 604 599]
 [590 604 594 ... 588 600 596]
 [597 601 594 ... 613 604 604]
 ...
 [595 598 578 ... 590 605 598]
 [603 596 595 ... 599 598 601]
 [594 596 588 ... 582 598 590]]
```

5 Visualising a bias frame

Make a Python script to convert a FITS file into a PNG image file. Here is an example.

Python Code 3: ao2021_s03_03.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.io.fits

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

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

# adding arguments
parser.add_argument ('-i', '--input', default='test.fits', \
                     help='intput FITS file')
parser.add_argument ('-o', '--output', default='test.png', \
                     help='output image file')

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

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

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

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

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

# primary HDU
hdu0 = hdu_list[0]

# reading header
```

```
header0 = hdu0.header

# reading data
data0 = hdu0.data

# closing FITS file
hdu_list.close ()

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

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

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

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

Run the script, and convert a FITS file into a PNG file.

```
% chmod a+x ao2021_s03_03.py
% ./ao2021_s03_03.py -h
usage: ao2021_s03_03.py [-h] [-i INPUT]

Reading a FITS file and calculating a mean

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        intput FITS file

% ./ao2021_s03_03.py -i data_ao2021_s03/lot_20210212_0079.fits -o 0079.png
data_ao2021_s03/lot_20210212_0079.fits ==> 0079.png
% ls -l 0079.png
-rw-r--r--  1 daisuke  taiwan  7442339 Mar  2 14:26 0079.png
% file 0079.png
0079.png: PNG image data, 2880 x 2160, 8-bit/color RGBA, non-interlaced
```

Show the image using your favourite image viewer software. (Fig. 2) For example, a command named "`feh`" can be used for this purpose.

```
% feh -dF 0079.png
```

If you do not have a command "`feh`" on your computer, and are willing to install it, visit the official web site of "`feh`". (Fig. 3)

- `https://feh.finalrewind.org/`

If you prefer to use GIMP, then try following.

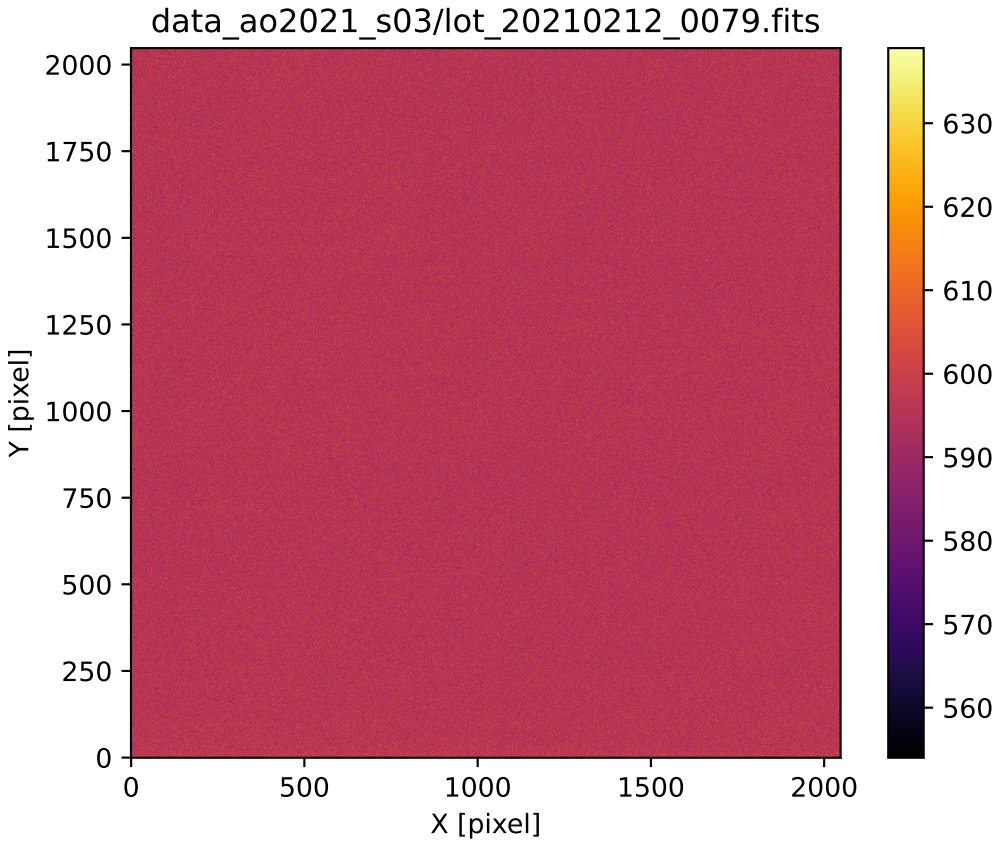

Figure 2: A visualised bias frame.

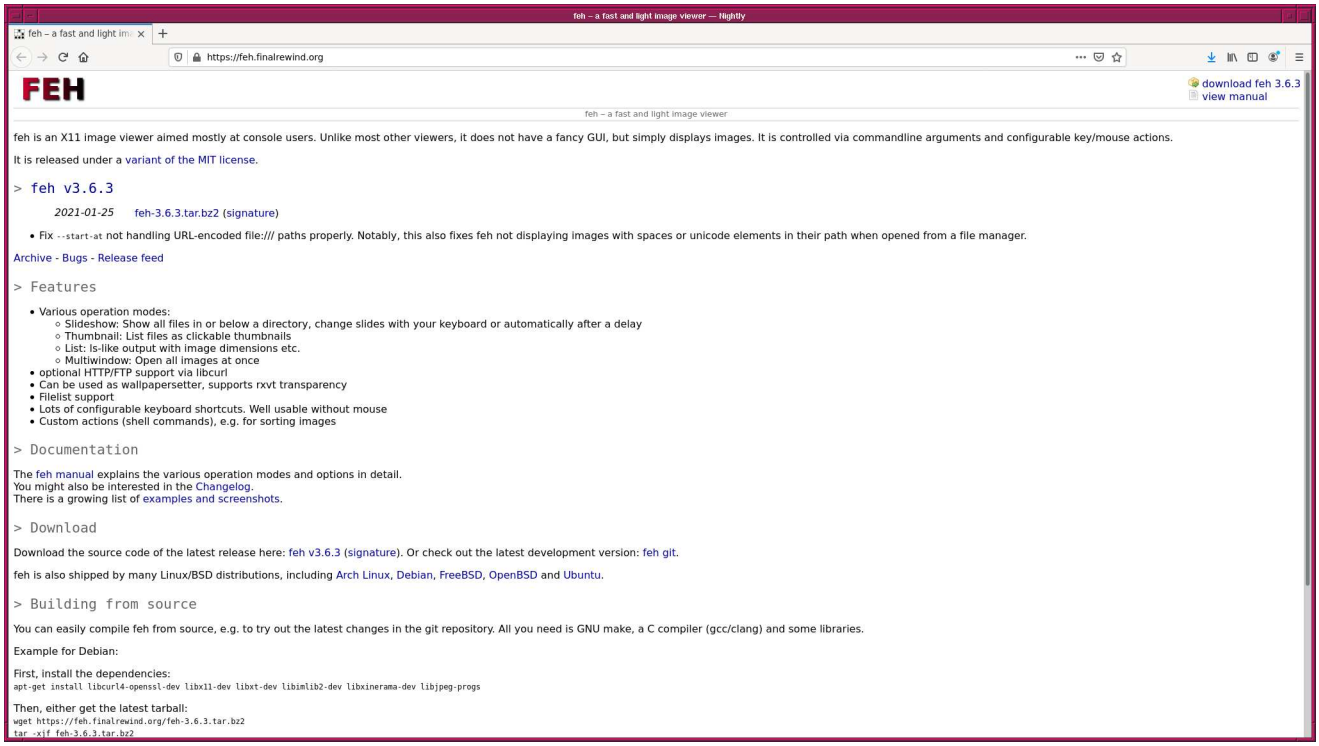

Figure 3: The official web page of the image viewer "`feh`".

```
% gimp 0079.png &
```

If you do not have a command "`gimp`" on your computer, and are willing to install it, visit the official web site of GIMP. (Fig. 4)

- `https://www.gimp.org/`

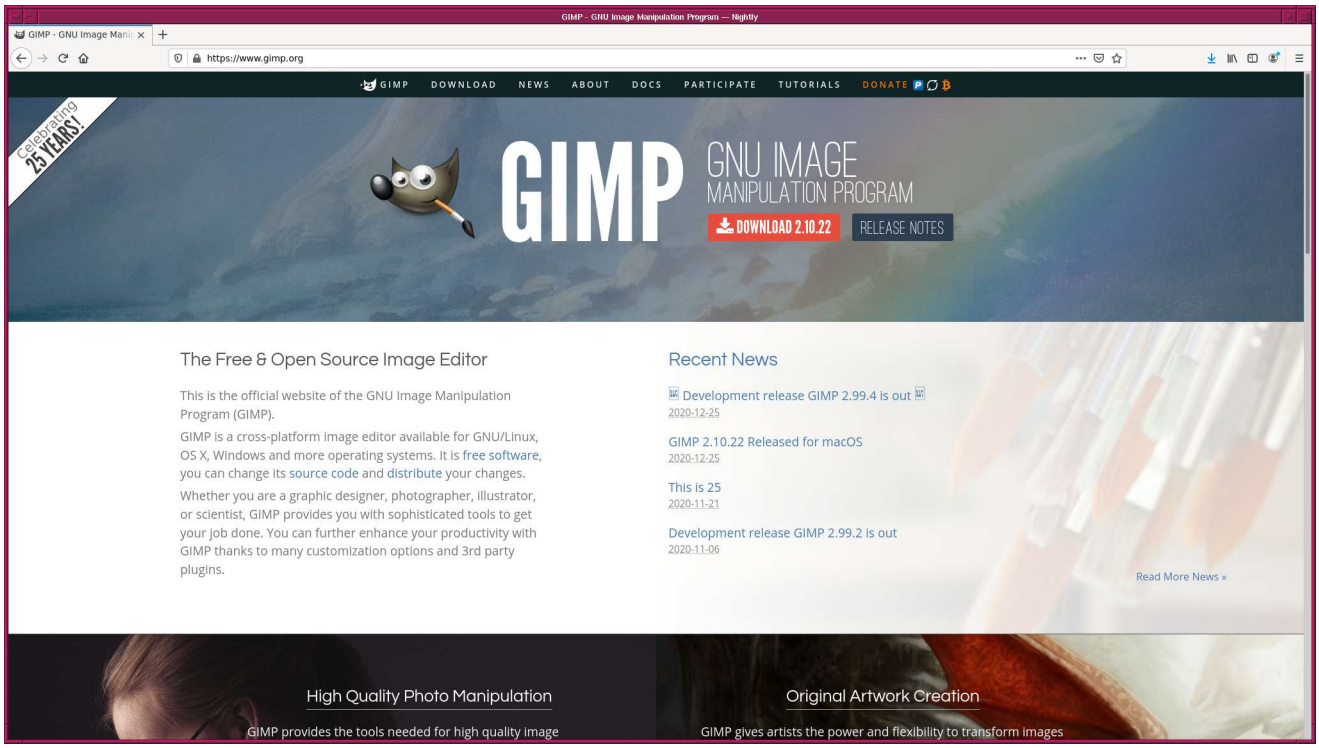

Figure 4: The official web page of the image manipulation program "GIMP".

6 Calculating a simple mean of a bias frame

Make a Python script to read a bias frame and calculate a simple mean.

Python Code 4: ao2021_s03_04.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing astropy module
import astropy.io.fits

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

# adding arguments
```

```
parser.add_argument ('-i', '--input', default='test.fits', \
                     help='intput FITS file')

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

# input FITS file
file_input = args.input

# 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 ()

# 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 ()

# a variable for a sum of all the pixel values
total = 0.0
# total number of pixels
n = data0.size

# calculating a simple mean
for i in range (len (data0) ):
    for j in range (len (data0[i]) ):
        total += data0[i][j]
mean = total / n

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

Execute the script, and calculate a mean of a bias frame.

```
% chmod a+x ao2021_s03_04.py
% ./ao2021_s03_04.py -h
usage: ao2021_s03_04.py [-h] [-i INPUT]

Reading a FITS file and calculating a mean

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        intput FITS file
```

```
% ./ao2021_s03_04.py -i data_ao2021_s03/lot_20210212_0079.fits
mean = 595.986696
```

The mean of all the pixel values of the file `lot_20210212_0079.fits` is 595.99.

7 Calculating mean, median, max, min, and stddev

In addition to a simple mean, calculate median, maximum value, minimum value, and standard deviation.

Python Code 5: ao2021_s03_05.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing math module
import math

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Reading a FITS file and calculating statistical values'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-i', '--input', default='test.fits', \
                     help='intput FITS file')

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

# input FITS file
file_input = args.input
filename = file_input.split ('/')[-1]

# 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 ()

# 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 ()

# initial value for maximum value
v_max = -9.99 * 10**10
# initial value for minimum value
v_min = +9.99 * 10**10
# a variable for a sum of all the pixel values
total = 0.0
# a variable for a square of sum of all the pixel values
total_sq = 0.0
# total number of pixels
n = data0.size
# a 1-dim. list for data
data0_1d = []

# calculating statistical values
for i in range (len (data0) ):
    for j in range (len (data0[i]) ):
        # for calculation of a mean
        total += data0[i][j]
        # for calculation of a variance
        total_sq += data0[i][j]**2
        # for maximum value
        if (data0[i][j] > v_max):
            v_max = data0[i][j]
        # for minimum value
        if (data0[i][j] < v_min):
            v_min = data0[i][j]
        # making 1-dim. list
        data0_1d.append (data0[i][j])

# mean
mean   = total / n
# variance
var    = total_sq / n - mean**2
# standard deviation
stddev = math.sqrt (var)

# sorting
data0_1d_sorted = sorted (data0_1d)
# median
if (n % 2 == 0):
    median = (data0_1d_sorted[int (n / 2) - 1] \
              + data0_1d_sorted[int (n / 2)]) / 2.0
else:
    median = data0_1d_sorted[int (n / 2)]

# printing result
print ("# file name, mean, median, stddev, min, max")
print ("%s : %f %f %f %f %f" \
       % (filename, mean, median, stddev, v_min, v_max) )
```

Run the script.

```
% ./ao2021_s03_05.py -i data_ao2021_s03/lot_20210212_0079.fits
# file name, mean, median, stddev, min, max
```

```
lot_20210212_0079.fits : 595.986696 596.000000 8.001158 554.000000 639.000000
```

The mean, median, stddev, min, and max for the file `lot_20210212_0079.fits` are 595.99, 596.00, 8.00, 554.00, and 639.00, respectively.

8 Calculating statistical values using Numpy

Use Numpy and calculate statistical values.

Python Code 6: ao2021_s03_06.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing math module
import math

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Reading a FITS file and calculating statistical values using Numpy'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-i', '--input', default='test.fits', \
                     help='intput FITS file')

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

# input FITS file
file_input = args.input
filename = file_input.split ('/')[-1]

# 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 ()

# 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 ()

# calculating statistical values
mean   = numpy.mean (data0)
median = numpy.median (data0)
stddev = numpy.std (data0)
v_min  = numpy.amin (data0)
v_max  = numpy.amax (data0)

# printing result
print ("# file name, mean, median, stddev, min, max")
print ("%s : %f %f %f %f %f" \
       % (filename, mean, median, stddev, v_min, v_max) )
```

Run the script and do the calculation.

```
% chmod a+x ao2021_s03_06.py
% ./ao2021_s03_06.py -h
usage: ao2021_s03_06.py [-h] [-i INPUT]

Reading a FITS file and calculating statistical values using Numpy

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        intput FITS file

% ./ao2021_s03_06.py -i data_ao2021_s03/lot_20210212_0079.fits
# file name, mean, median, stddev, min, max
lot_20210212_0079.fits : 595.986696 596.000000 8.001158 554.000000 639.000000
```

Compare the results of calculations by `ao2021_s03_05.py` and `ao2021_s03_06.py`.

```
% ./ao2021_s03_05.py -i data_ao2021_s03/lot_20210212_0079.fits > s03_05.data
% ./ao2021_s03_06.py -i data_ao2021_s03/lot_20210212_0079.fits > s03_06.data
% diff s03_05.data s03_06.data
```

Two Python scripts give the exactly the same results.
Try the script for the file `lot_20210212_0084.fits`.

```
% ./ao2021_s03_06.py -i data_ao2021_s03/lot_20210212_0084.fits
# file name, mean, median, stddev, min, max
lot_20210212_0084.fits : 595.101543 595.000000 8.044479 555.000000 2202.000000
```

A pixel of the image `lot_20210212_0084.fits` has an anomalously high pixel value of 2202.00. We should think of more robust way to calculate a mean by rejecting outliers.

9 Constructing a histogram

Make a Python script to construct a histogram of pixel values of a bias frame.

Python Code 7: ao2021_s03_07.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing math module
import math

# importing numpy module
import numpy

# 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 constructing a histogram'
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='outtput image file')
parser.add_argument ('-a', '--min', type=float, default=400.0, \
                     help='minimum value for histogram')
parser.add_argument ('-b', '--max', type=float, default=800.0, \
                     help='maximum value for histogram')
parser.add_argument ('-w', '--width', type=int, default=1.0, \
                     help='width of a bin for histogram')

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

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

# parameters
a = args.min
b = args.max
width = args.width
nbin = int ( (b - a) / width ) + 1

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

```
    # exit
    sys.exit ()

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

# if a >= b, then skip
if (a >= b):
    # printing a message
    print ("maximum value \"a\" must be greater than minimum value \"b\".")
    # exit
    sys.exit ()

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

# primary HDU
hdu0 = hdu_list[0]

# reading header
header0 = hdu0.header

# reading data
data0 = hdu0.data

# closing FITS file
hdu_list.close ()

# initialisation of Numpy arrays for histogram
histogram_x = numpy.linspace (a, b, nbin)
histogram_y = numpy.zeros (nbin, dtype='int64')

# making a histogram
for i in range ( len (data0) ):
    for j in range ( len (data0[i]) ):
        # skipping data if it is outside the range [a, b]
        if ( (data0[i][j] > b) or (data0[i][j] < a) ):
            continue
        # counting
        histogram_y[int ( (data0[i][j] - a) / width)] += 1

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

# labels
label_x = 'Pixel Value [ADU]'
label_y = 'Number of Pixels'
ax.set_xlabel (label_x)
ax.set_ylabel (label_y)

# plotting histogram
ax.bar (histogram_x, histogram_y, width, edgecolor='black', \
```

```
        linewidth=0.3, align='edge', label='Pixel values')
ax.legend ()

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

Run the script.

```
% chmod a+x ao2021_s03_07.py
% ./ao2021_s03_07.py -h
usage: ao2021_s03_07.py [-h] [-i INPUT] [-o OUTPUT] [-a MIN] [-b MAX]
                        [-w WIDTH]

Reading a FITS file and constructing a histogram

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        intput FITS file
  -o OUTPUT, --output OUTPUT
                        outtput image file
  -a MIN, --min MIN     minimum value for histogram
  -b MAX, --max MAX     maximum value for histogram
  -w WIDTH, --width WIDTH
                        width of a bin for histogram

% ./ao2021_s03_07.py -i data_ao2021_s03/lot_20210212_0079.fits -o hist_0079.png
% ls -l hist_0079.png
-rw-r--r--  1 daisuke  taiwan  108530 Mar  2 16:55 hist_0079.png
```

Display the image. (Fig. 5)

```
% feh -dF hist_0079.png
```

Adjust the range, and run the script again.

```
% ./ao2021_s03_07.py -i data_ao2021_s03/lot_20210212_0079.fits \
? -o hist_0079_2.png -a 550 -b 650
% ls -l hist_0079_2.png
-rw-r--r--  1 daisuke  taiwan  108717 Mar  2 17:00 hist_0079_2.png
```

Display the image. (Fig. 6)

```
% feh -dF hist_0079_2.png
```

Almost all the pixels have pixel values between $(\bar{x} - 30)$ and $(\bar{x} + 30)$ ADU which roughly correspond $\bar{x} \pm 4\sigma$.

10 Calculating a mean using sigma clipping

More robust estimate of a mean is possible by using sigma clipping algorithm. Make a Python script to calculate a mean using sigma clipping algorithm.

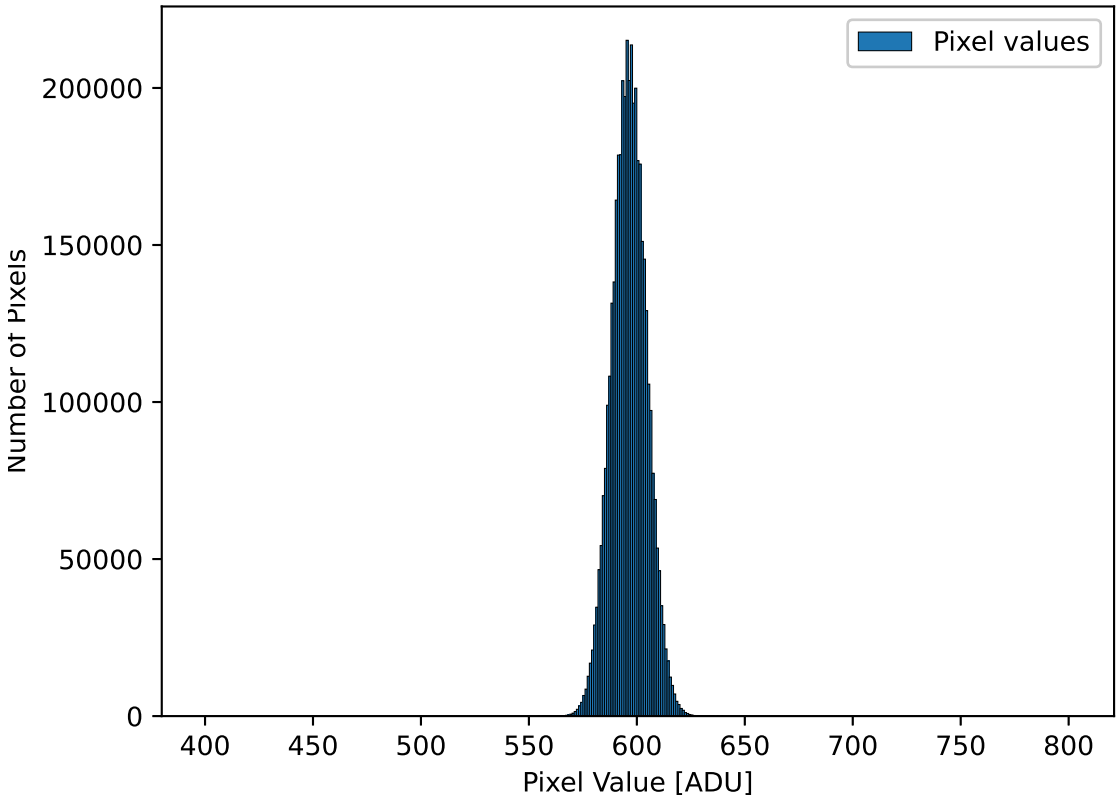

Figure 5: The histogram of pixel values of a bias frame `lot_20210212_0079.fits`.

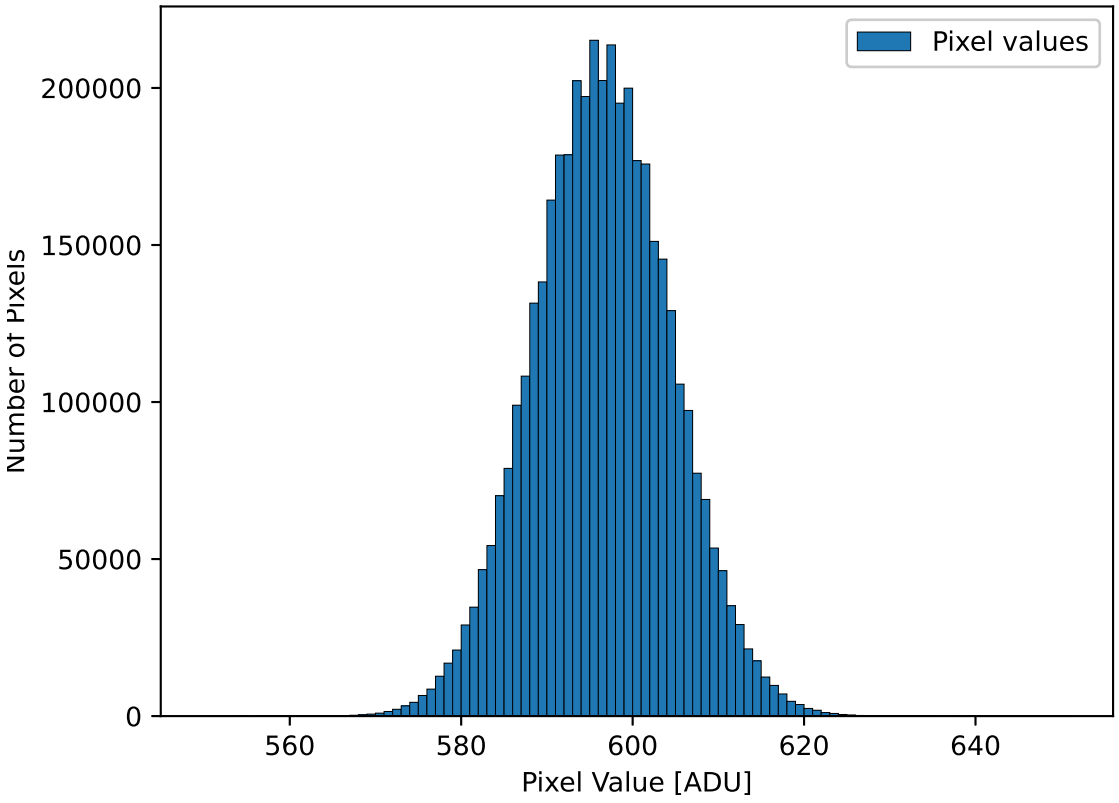

Figure 6: The updated histogram of pixel values of a bias frame `lot_20210212_0079.fits`.

Python Code 8: ao2021_s03_08.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing math module
import math

# importing astropy module
import astropy.io.fits

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

# adding arguments
parser.add_argument ('-i', '--input', default='test.fits', \
                     help='intput FITS file')
parser.add_argument ('-s', '--sigma', type=float, default=5.0, \
                     help='factor for sigma clipping')

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

# input FITS file
file_input = args.input
filename = file_input.split ('/')[-1]
nsigma = args.sigma

# 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 ()

# 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 ()

# initial value for maximum value
v_max         = -9.99 * 10**10
v_max_clipped = -9.99 * 10**10
# initial value for minimum value
```

```
v_min         = +9.99 * 10**10
v_min_clipped = +9.99 * 10**10
# a variable for a sum of all the pixel values
total         = 0.0
total_clipped = 0.0
# a variable for a square of sum of all the pixel values
total_sq         = 0.0
total_sq_clipped = 0.0
# total number of pixels
n = data0.size

# calculating statistical values
for row in data0:
    for x in row:
        # for calculation of a mean
        total += x
        # for calculation of a variance
        total_sq += x**2
        # for maximum value
        if (x > v_max):
            v_max = x
        # for minimum value
        if (x < v_min):
            v_min = x

# mean
mean   = total / n
# variance
var    = total_sq / n - mean**2
# standard deviation
stddev = math.sqrt (var)

# lists for rejected and accepted values
rejected = []
accepted = []

# calculating statistical values
for row in data0:
    for x in row:
        # if the pixel value is outside of
        # [mean-nsigma*stddev,mean+nsigma*stddev], then reject
        if ( (x > mean + nsigma * stddev) or (x < mean - nsigma * stddev) ):
            # appending the pixel value to the list "rejected"
            rejected.append (x)
        else:
            # appending the pixel value to the list "accepted"
            accepted.append (x)
            # for calculation of a mean
            total_clipped += x
            # for calculation of a variance
            total_sq_clipped += x**2
            # for maximum value
            if (x > v_max_clipped):
                v_max_clipped = x
            # for minimum value
            if (x < v_min_clipped):
                v_min_clipped = x

# numbers of accepted and rejected pixels
```

```
n_accepted = len (accepted)
n_rejected = len (rejected)

# mean
mean_clipped   = total_clipped / n_accepted
# variance
var_clipped    = total_sq_clipped / n_accepted - mean_clipped**2
# standard deviation
stddev_clipped = math.sqrt (var_clipped)

# printing result
print ("before clipping:")
print ("  %s : %f %f %f %f" % (filename, mean, stddev, v_min, v_max) )
print ("after clipping:")
print ("  %s : %f %f %f %f" % (filename, mean_clipped, stddev_clipped, \
                               v_min_clipped, v_max_clipped) )
print ("  number of accepted pixels =", n_accepted)
print ("  number of rejected pixels =", n_rejected)
print ("  rejected pixel values =", rejected)
```

Execute the script.

```
% chmod a+x ao2021_s03_08.py
% ./ao2021_s03_08.py -h
usage: ao2021_s03_08.py [-h] [-i INPUT] [-s SIGMA]

Calculating statistical values using sigma clipping

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        intput FITS file
  -s SIGMA, --sigma SIGMA
                        factor for sigma clipping

% ./ao2021_s03_08.py -i data_ao2021_s03/lot_20210212_0084.fits
before clipping:
  lot_20210212_0084.fits : 595.101543 8.044479 555.000000 2202.000000
after clipping:
  lot_20210212_0084.fits : 595.100994 8.003903 555.000000 635.000000
  number of accepted pixels = 4194297
  number of rejected pixels = 7
  rejected pixel values = [640, 646, 2202, 942, 712, 691, 637]
```

11 Sigma clipping using Astropy

Use Astropy to carry out sigma clipping.

Python Code 9: ao2021_s03_09.py

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

# importing argparse module
import argparse

# importing sys module
import sys
```

```
# importing math module
import math

# importing numpy module
import numpy

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

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

# adding arguments
parser.add_argument ('-i', '--input', default='test.fits', \
                     help='intput FITS file')
parser.add_argument ('-s', '--sigma', type=float, default=5.0, \
                     help='factor for sigma clipping')
parser.add_argument ('-n', '--niter', type=int, default=10, \
                     help='maximum number of iterations')

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

# input FITS file
file_input = args.input
filename   = file_input.split ('/')[-1]
nsigma     = args.sigma
niter      = args.niter

# 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 ()

# 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 ()

# simple mean
mean   = numpy.mean (data0)
stddev = numpy.std (data0)
v_min  = numpy.amin (data0)
v_max  = numpy.amax (data0)
```

```
# sigma clipped mean
data0_sigclip  = astropy.stats.sigma_clip (data0, sigma=nsigma, \
                                           maxiters=niter, masked=False)
mean_sigclip   = numpy.mean (data0_sigclip)
stddev_sigclip = numpy.std (data0_sigclip)
v_min_sigclip  = numpy.amin (data0_sigclip)
v_max_sigclip  = numpy.amax (data0_sigclip)

# printing result
print ("before clipping:")
print ("  %s : %f %f %f %f" % (filename, mean, stddev, v_min, v_max) )
print ("after clipping:")
print ("  %s : %f %f %f %f" % (filename, mean_sigclip, stddev_sigclip, \
                              v_min_sigclip, v_max_sigclip) )
print ("size of data0             =", data0.size)
print ("size of data0_sigclip     =", data0_sigclip.size)
print ("number of rejected pixels =", data0.size - data0_sigclip.size)
```

Run the script, and compare the results with those calculated by the previous script.

```
% chmod a+x ao2021_s03_09.py
% ./ao2021_s03_09.py -h
usage: ao2021_s03_09.py [-h] [-i INPUT] [-s SIGMA] [-n NITER]

Calculating statistical values using sigma clipping

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        intput FITS file
  -s SIGMA, --sigma SIGMA
                        factor for sigma clipping
  -n NITER, --niter NITER
                        maximum number of iterations

% ./ao2021_s03_09.py -i data_ao2021_s03/lot_20210212_0084.fits
before clipping:
  lot_20210212_0084.fits : 595.101543 8.044479 555.000000 2202.000000
after clipping:
  lot_20210212_0084.fits : 595.100994 8.003903 555.000000 635.000000
size of data0             = 4194304
size of data0_sigclip     = 4194297
number of rejected pixels = 7
```

12 A convenient way to give a rough estimate of readout noise

Here is a convenient way to give a crude estimate of readout noise. First, we choose two bias frames. Second, we subtract one bias frame from the other. Third, we calculate the standard deviation of difference of two bias frames. Then, a crude estimate of readout noise is given by dividing the standard deviation by $\sqrt{2}$.

Python Code 10: ao2021_s03_10.py

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

# importing argparse module
import argparse
```

```
# importing sys module
import sys

# importing math module
import math

# importing numpy module
import numpy

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

# construction of parser object
desc = 'estimating readout noise from two bias frames'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-b1', '--bias1', default='bias1.fits', \
                     help='bias frame 1')
parser.add_argument ('-b2', '--bias2', default='bias2.fits', \
                     help='bias frame 2')
parser.add_argument ('-s', '--sigma', type=float, default=5.0, \
                     help='factor for sigma clipping')
parser.add_argument ('-n', '--niter', type=int, default=10, \
                     help='maximum number of iterations')

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

# input FITS file
file_bias1 = args.bias1
file_bias2 = args.bias2
filename1  = file_bias1.split ('/')[-1]
filename2  = file_bias2.split ('/')[-1]
nsigma     = args.sigma
niter      = args.niter

# if input file is not a FITS file, then skip
if not ( (file_bias1[-5:] == '.fits') and (file_bias2[-5:] == '.fits') ):
    # printing a message
    print ("Error: input file must be a FITS file!")
    # exit
    sys.exit ()

# opening FITS file
hdu_list1 = astropy.io.fits.open (file_bias1)
hdu_list2 = astropy.io.fits.open (file_bias2)

# primary HDU
hdu1 = hdu_list1[0]
hdu2 = hdu_list2[0]

# reading header
header1 = hdu1.header
header2 = hdu2.header

# reading data and conversion from uint16 into float64
```

```
bias1 = hdu1.data.astype (numpy.float64)
bias2 = hdu2.data.astype (numpy.float64)

# closing FITS file
hdu_list1.close ()
hdu_list2.close ()

# calculation of (bias1 - bias2)
diff_b1_b2 = bias1 - bias2

# sigma clipped mean and stddev
diff_sigclip  = astropy.stats.sigma_clip (diff_b1_b2, sigma=nsigma, \
                                          maxiters=niter, masked=False)
mean_sigclip   = numpy.mean (diff_sigclip)
stddev_sigclip = numpy.std  (diff_sigclip)
v_min_sigclip  = numpy.amin (diff_sigclip)
v_max_sigclip  = numpy.amax (diff_sigclip)

# printing result
print ("selected bias frames: %s and %s" % (filename1, filename2) )
print ("mean of bias1 = %f" % numpy.mean (bias1) )
print ("mean of bias2 = %f" % numpy.mean (bias2) )
print ("mean of diff = %f" % numpy.mean (diff_b1_b2) )
print ("mean_sigclip = %f" % mean_sigclip)
print ("stddev of (%s - %s) = %f ADU" \
       % (filename1, filename2, stddev_sigclip) )
print ("estimated readout noise = %f ADU" % (stddev_sigclip / math.sqrt(2.0) ) )
```

Execute the script, and calculate readout noise.

```
% chmod a+x ao2021_s03_10.py
% ./ao2021_s03_10.py -b1 data_ao2021_s03/lot_20210212_0079.fits \
? -b2 data_ao2021_s03/lot_20210212_0080.fits
selected bias frames: lot_20210212_0079.fits and lot_20210212_0080.fits
mean of bias1 = 595.986696
mean of bias2 = 596.661863
mean of diff = -0.675167
mean_sigclip = -0.675066
stddev of (lot_20210212_0079.fits - lot_20210212_0080.fits) = 11.310364 ADU
estimated readout noise = 7.997635 ADU
```

Here is the other example.

```
% ./ao2021_s03_10.py -b1 data_ao2021_s03/lot_20210212_0081.fits \
? -b2 data_ao2021_s03/lot_20210212_0082.fits
selected bias frames: lot_20210212_0081.fits and lot_20210212_0082.fits
mean of bias1 = 595.912266
mean of bias2 = 595.532202
mean of diff = 0.380064
mean_sigclip = 0.380037
stddev of (lot_20210212_0081.fits - lot_20210212_0082.fits) = 11.291145 ADU
estimated readout noise = 7.984045 ADU
```

13 Time variation of mean bias level

Make a Python script to check the time variation of mean bias level.

Python Code 11: ao2021_s03_11.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing math module
import math

# 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

# construction of parser object
desc = 'Examining time variation of mean bias level'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('files', nargs='+', help='bias frames')
parser.add_argument ('-s', '--sigma', type=float, default=5.0, \
                     help='factor for sigma clipping')
parser.add_argument ('-n', '--niter', type=int, default=10, \
                     help='maximum number of iterations')
parser.add_argument ('-o', '--output', default='bias.png', \
                     help='output file name')

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

# input FITS file
files       = args.files
nsigma      = args.sigma
niter       = args.niter
file_output = args.output

# data
data_datetime = numpy.array ([], dtype='datetime64[ms]')
data_mean     = numpy.array ([], dtype='float64')
data_stddev   = numpy.array ([], dtype='float64')

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

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

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

# processing files
for file_fits in files:
    print ("Now processing the file \"%s\"..." % file_fits)

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

    # primary HDU
    hdu0 = hdu_list[0]

    # reading header
    header0 = hdu0.header

    # reading data and conversion from uint16 into float64
    data0 = hdu0.data.astype (numpy.float64)

    # closing FITS file
    hdu_list.close ()

    # date/time
    date = header0['DATE-OBS']
    time = header0['TIME-OBS']
    datetime_str = "%sT%s" % (date, time)
    datetime64 = numpy.datetime64 (datetime_str)

    # sigma clipped mean and stddev
    data0_sigclip  = astropy.stats.sigma_clip (data0, sigma=nsigma, \
                                               maxiters=niter, masked=False)
    mean_sigclip   = numpy.mean (data0_sigclip)
    stddev_sigclip = numpy.std  (data0_sigclip)

    # appending data to the lists
    data_datetime = numpy.append (data_datetime, datetime64)
    data_mean     = numpy.append (data_mean, mean_sigclip)
    data_stddev   = numpy.append (data_stddev, stddev_sigclip)

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

# labels
label_x = 'Date/Time [UT]'
label_y = 'Mean Bias Level [ADU]'
ax.set_xlabel (label_x)
ax.set_ylabel (label_y)
ax.xaxis.set_major_formatter (matplotlib.dates.DateFormatter ('%H:%M'))

# axis settings
y_min = 580.0
y_max = 610.0
ax.set_ylim (y_min, y_max)
```

```
# plotting data
ax.errorbar (data_datetime, data_mean, yerr=data_stddev, \
             marker='o', color='blue', markersize=3, linestyle='none', \
             ecolor='black', capsize=2, \
             label='Mean bias level')
ax.legend ()

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

Run the script, and generate a PNG file.

```
% chmod a+x ao2021_s03_11.py
% ./ao2021_s03_11.py -o biaslevel.png data_ao2021_s03/*.fits
Now processing the file "data_ao2021_s03/lot_20210212_0079.fits"...
Now processing the file "data_ao2021_s03/lot_20210212_0080.fits"...
Now processing the file "data_ao2021_s03/lot_20210212_0081.fits"...
Now processing the file "data_ao2021_s03/lot_20210212_0082.fits"...
Now processing the file "data_ao2021_s03/lot_20210212_0083.fits"...

.....

Now processing the file "data_ao2021_s03/lot_20210212_0814.fits"...
Now processing the file "data_ao2021_s03/lot_20210212_0815.fits"...
Now processing the file "data_ao2021_s03/lot_20210212_0816.fits"...
Now processing the file "data_ao2021_s03/lot_20210212_0817.fits"...
Now processing the file "data_ao2021_s03/lot_20210212_0818.fits"...
% ls -l biaslevel.png
-rw-r--r--  1 daisuke  taiwan  149321 Mar  2 23:49 biaslevel.png
```

Show the PNG image using an image viewer program. (Fig. 7)

```
% feh -dF biaslevel.png
```

Roughly speaking, the bias level seems to be stable over time.

14 For your training

- Read chapter 4 of "Handbook of CCD Astronomy" and learn about bias 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 Astropy, and learn about sigma clipping algorithm.
  - `https://docs.astropy.org/en/stable/stats/index.html`
  - `https://docs.astropy.org/en/stable/stats/robust.html`

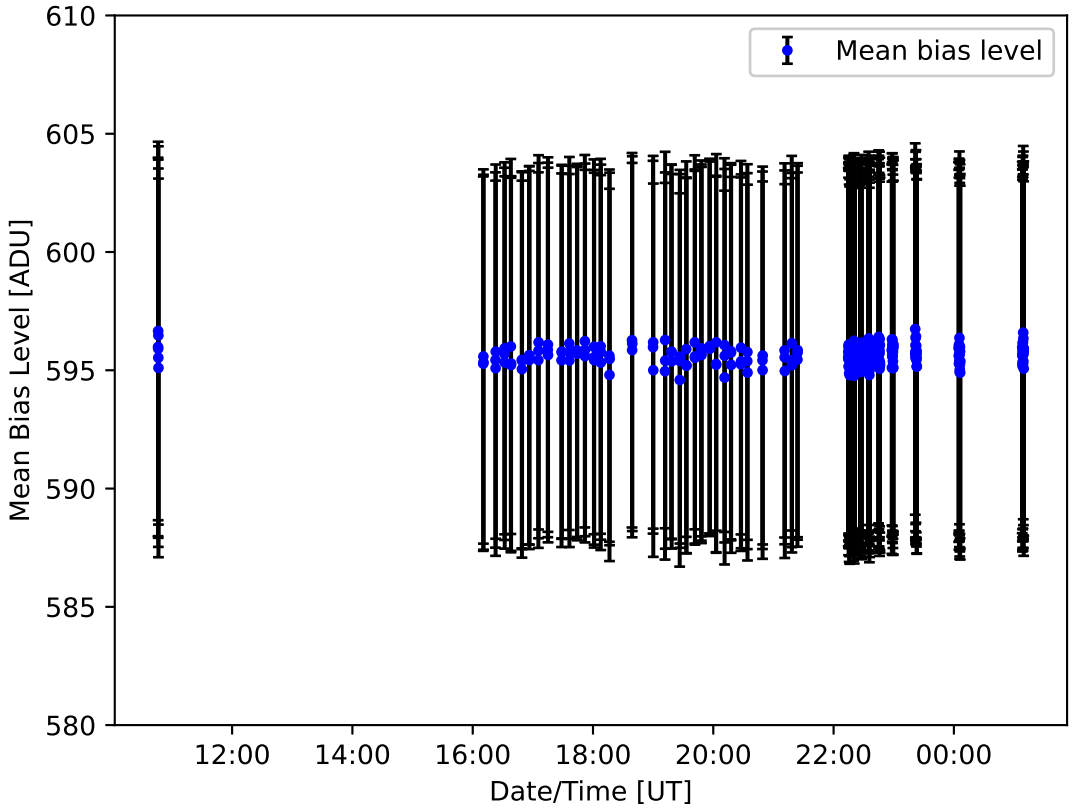

Figure 7: Time variation of mean bias level.

15 Assignment

1. What is bias? Describe bias. How to take bias frames? Why do we need to take bias frames during the observing run? Why mean value of bias is not zero? How useful and important are bias frames?

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

3. Choose three bias frames. Make your own Python script to make histograms of those three bias 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 bias frames. Make your own Python script to calculate mean, median, variance, standard deviation, minimum value, maximum value of those three bias 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 bias frames. Make your own Python script to calculate mean, median, variance, standard deviation, minimum value, maximum value of those three bias 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 bias 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 bias frame.

7. What is sigma clipping algorithm? How useful is it? Give some examples that we should use sigma clipping. Describe why sigma clipping is more robust.

8. Other than sigma clipping algorithm, is there any other robust estimator? If so, explain the method.

9. Make your own function which works like `astropy.stats.sigma_clip ()`. Show the source code of your function. Test your function with some bias data for this session. Take a screenshot of your computer display to show the results.

10. Make your own Python script which works like `imstatistics` task of IRAF (Image Reduction and Analysis Facility). Implement sigma clipping for your Python script. Show the source code of your function. Test your script with some bias data for this session. Take a screenshot of your computer display to show the results.

    - `https://iraf.net/irafhelp.php?val=imstatistics&help=Help+Page`

11. To estimate readout noise, we first subtracted one bias frame from another. Describe why we did this. Then, we estimated standard deviation of difference of two bias frames and divided standard deviation by $\sqrt{2}$. Describe why we divide the standard deviation by $\sqrt{2}$.

12. Choose two bias frames. Make a Python script to subtract one bias from another. Visualise difference of two bias frames using Matplotlib. Show the image you have produced.

13. Make your own Python script to show time variation of mean bias level. Use Matplotlib to produce a plot. Show the source code of your Python script and plot you have produced. Discuss the time variability of bias frames.