Advanced Astronomical Observations 2021
Session 17: Background Estimation

Kinoshita Daisuke

21 May 2021
publicly accessible version

About this file...

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

For this session, we try background estimation.

1 Python scripts for this session

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

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

```
% git clone https://github.com/kinoshitadaisuke/ncu_advobs_202102.git
Cloning into 'ncu_advobs_202102'...
remote: Enumerating objects: 68, done.
remote: Counting objects: 100% (68/68), done.
remote: Compressing objects: 100% (64/64), done.
remote: Total 68 (delta 31), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (68/68), 34.48 KiB | 882.00 KiB/s, done.
Resolving deltas: 100% (31/31), done.
% ls
ncu_advobs_202102/
% ls -l ncu_advobs_202102/
total 1
-rw-r--r--  1 daisuke  wheel  35149 May 21 01:25 LICENSE
-rw-r--r--  1 daisuke  wheel    362 May 21 01:25 README
drwxr-xr-x  2 daisuke  wheel    512 May 21 01:25 s16_skybackground/
drwxr-xr-x  2 daisuke  wheel    512 May 21 01:25 s17_backgroundestimation/
% ls -l ncu_advobs_202102/s17_backgroundestimation/
total 1
```

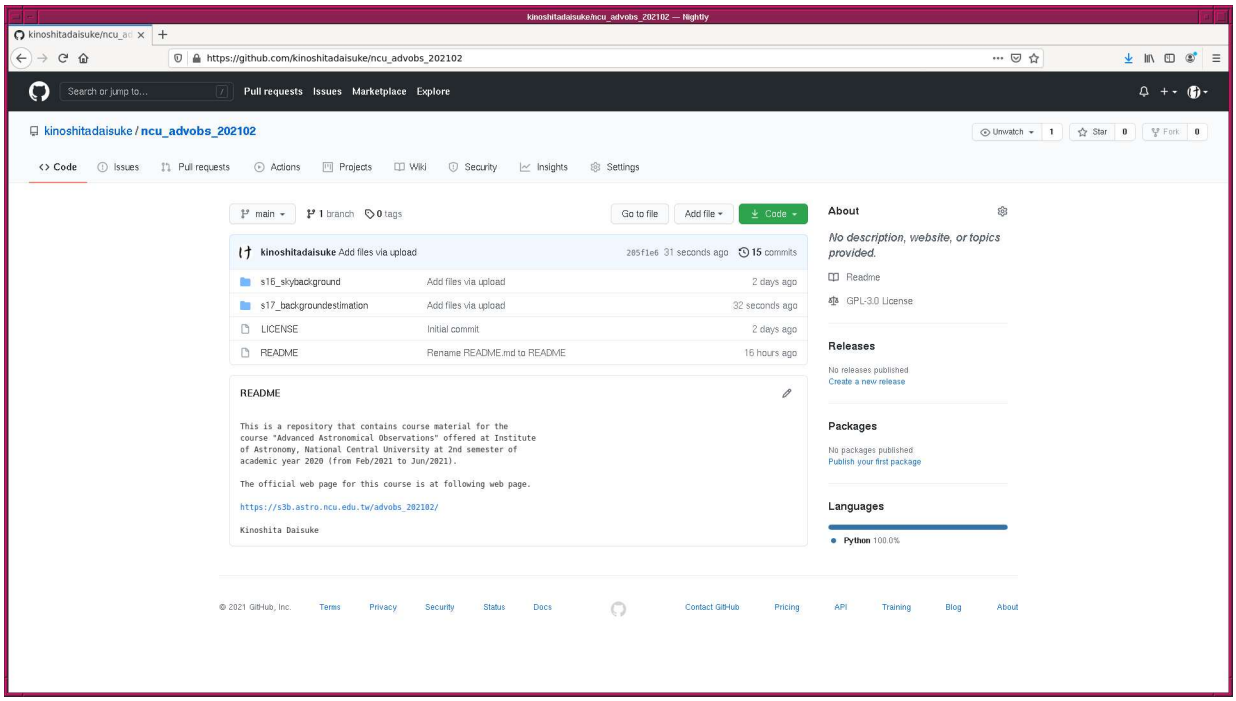

Figure 1: The GitHub repository for Python scripts for this course.

```
-rw-r--r--  1 daisuke  wheel    35 May 21 01:25 README
-rw-r--r--  1 daisuke  wheel  1571 May 21 01:25 ao2021_s17_01.py
-rw-r--r--  1 daisuke  wheel  4226 May 21 01:25 ao2021_s17_02.py
-rw-r--r--  1 daisuke  wheel  6409 May 21 01:25 ao2021_s17_03.py
-rw-r--r--  1 daisuke  wheel  3144 May 21 01:25 ao2021_s17_04.py
-rw-r--r--  1 daisuke  wheel  2782 May 21 01:25 ao2021_s17_05.py
% head -15 ncu_advobs_202102/s17_backgroundestimation/ao2021_s17_01.py
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2021/05/20 22:57:04 (CST) daisuke>
#

# importing numpy module
import numpy
import numpy.random

# importing astropy module
import astropy.table

# generating a new astropy table
table_stars = astropy.table.Table ()
```

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

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

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

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

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

2 Generating synthetic data

2.1 Preparing an Astropy table

Make a Python script to generate an Astropy table.

Python Code 1: ao2021_s17_01.py

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

# importing numpy module
import numpy
import numpy.random

# importing astropy module
import astropy.table

# generating a new astropy table
table_stars = astropy.table.Table ()

# number of stars to generate
nstars = 100

# flux of faintest stars
flux_min = 1000.0

# image size
image_size_x = 2048
image_size_y = 2048

# FWHM of PSF
fwhm_x = 3.5
fwhm_y = 3.5
fwhm_stddev_x = 0.1
fwhm_stddev_y = 0.1

# generating random numbers
position_x = numpy.random.default_rng ().uniform (0, image_size_x, nstars)
position_y = numpy.random.default_rng ().uniform (0, image_size_y, nstars)
theta_deg  = numpy.random.default_rng ().uniform (0, 360, nstars)
psf_x      = numpy.random.default_rng ().normal (loc=fwhm_x, \
                                                 scale=fwhm_stddev_x, \
                                                 size=nstars)
psf_y      = numpy.random.default_rng ().normal (loc=fwhm_y, \
                                                 scale=fwhm_stddev_y, \
                                                 size=nstars)
powerlaw   = numpy.random.default_rng ().power (1.5, size=nstars)
flux       = flux_min / powerlaw

# conversion from degree to radian
theta_rad = numpy.radians (theta_deg)

# adding data to the table
```

```
table_stars['amplitude'] = flux
table_stars['x_mean']    = position_x
table_stars['y_mean']    = position_y
table_stars['x_stddev']  = psf_x
table_stars['y_stddev']  = psf_y
table_stars['theta']     = theta_rad

# printing table
print (table_stars)
```

Execute the script.

```
% chmod a+x ao2021_s17_01.py
% ./ao2021_s17_01.py
    amplitude             x_mean        ...       y_stddev             theta
----------------- ------------------ ... ------------------ ------------------
1169.6611154922923 1767.4926206096886 ... 3.5372009075980517 1.8415073590251312
2238.2041040721474  475.2038672034089 ... 3.5403957913628252 0.7608346930515217
1331.0746956810553  110.2029294584238 ...  3.415492875738168 4.1923417364289834
1213.4168870731412  901.7813044178613 ... 3.3729833598334986  5.133070821083168
1206.5954813462365  657.8325819356166 ... 3.5713688212546733  1.831550298987964
11596.695514147952  322.6574730673483 ...  3.441329496215634 0.9957040260896738
 13563.25987798269  813.9554362156002 ... 3.6256707325503696 1.0092946014689839
               ...                ... ...                ...                ...
1107.3108309432725  200.9024618775229 ... 3.3902867263996184  5.636738683795345
  1439.13087287991  871.0992313500176 ...  3.636456847103764  2.455085602642021
1031.4952095186104   825.868681150677 ... 3.4775687325105715 2.1236893672965023
1744.7463324933667  976.1502120617004 ... 3.2793467288217686 1.8793498401258761
 1126.469199362587  974.9802651240152 ... 3.6588522422528023 3.3620960357595733
1628.9963296560813 1345.6630995019689 ...  3.477925421717793  2.253719711107519
1086.6166076761622  851.7228896199408 ... 3.3484518689588634  4.681967113975408
Length = 100 rows
```

2.2 Generating a synthetic image with 200 artificial stars

Make a Python script to generate a synthetic image with 200 artificial stars.

Python Code 2: ao2021_s17_02.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing numpy module
import numpy
import numpy.random

# importing astropy module
import astropy.table

# importing photutils module
import photutils.datasets

# constructing parser object
```

```
desc = 'generating a synthetic image with artificial stars'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-x', '--size-x', type=int, default=2048, \
                     help='image size in X-axis (default: 2048)')
parser.add_argument ('-y', '--size-y', type=int, default=2048, \
                     help='image size in Y-axis (default: 2048)')
parser.add_argument ('-n', '--nstars', type=int, default=100, \
                     help='number of stars to generate (default: 100)')
parser.add_argument ('-f', '--flux-min', type=float, default=1000.0, \
                     help='minimum flux of stars (default: 1000)')
parser.add_argument ('-p', '--fwhm-psf', type=float, default=3.5, \
                     help='FWHM of PSF in pixel (default: 3.5)')
parser.add_argument ('-d', '--fwhm-stddev', type=float, default=0.1, \
                     help='stddev of FWHM distribution in pixel (default: 0.1)')
parser.add_argument ('-s', '--sky', type=float, default=1000.0, \
                     help='sky background level in ADU (default: 1000)')
parser.add_argument ('-e', '--sky-stddev', type=float, default=30.0, \
                     help='stddev of sky background in ADU (default: 30)')
parser.add_argument ('-o', '--output-file', default='', \
                     help='output file name')

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

# image size
image_size_x = args.size_x
image_size_y = args.size_y
image_shape  = (image_size_x, image_size_y)

# number of stars to generate
nstars = args.nstars

# flux of faintest stars
flux_min = args.flux_min

# FWHM of PSF
fwhm_x = args.fwhm_psf
fwhm_y = args.fwhm_psf
fwhm_stddev_x = args.fwhm_stddev
fwhm_stddev_y = args.fwhm_stddev

# sky background level and stddev
sky_mean   = args.sky
sky_stddev = args.sky_stddev

# output file name
file_output = args.output_file

# check of output file name
if not (file_output[-5:] == '.fits'):
    print ("Output file must be a FITS file.")
    sys.exit ()

# generating a new astropy table
table_stars = astropy.table.Table ()

# generating random numbers
```

```
position_x = numpy.random.default_rng().uniform(0, image_size_x, nstars)
position_y = numpy.random.default_rng().uniform(0, image_size_y, nstars)
theta_deg  = numpy.random.default_rng().uniform(0, 360, nstars)
psf_x      = numpy.random.default_rng().normal(loc=fwhm_x, \
                                               scale=fwhm_stddev_x, \
                                               size=nstars)
psf_y      = numpy.random.default_rng().normal(loc=fwhm_y, \
                                               scale=fwhm_stddev_y, \
                                               size=nstars)
powerlaw   = numpy.random.default_rng().power(1.5, size=nstars)
flux       = flux_min / powerlaw

# conversion from degree to radian
theta_rad = numpy.radians(theta_deg)

# adding data to the table
table_stars['amplitude'] = flux
table_stars['x_mean']    = position_x
table_stars['y_mean']    = position_y
table_stars['x_stddev']  = psf_x
table_stars['y_stddev']  = psf_y
table_stars['theta']     = theta_rad

# generating stars
image_stars = photutils.datasets.make_gaussian_sources_image(image_shape, \
                                                             table_stars)

# generating sky background
image_sky = photutils.datasets.make_noise_image(image_shape, \
                                                distribution='gaussian', \
                                                mean=sky_mean, \
                                                stddev=sky_stddev)
# generating synthetic image
image = image_stars + image_sky

# making WCS
wcs = photutils.datasets.make_wcs(image_shape)

# making Image HDU
hdu = photutils.datasets.make_imagehdu(image, wcs=wcs)

# writing a FITS file
hdu.writeto(file_output)
```

Run the script, and generate a synthetic image.

```
% chmod a+x ao2021_s17_02.py
% ./ao2021_s17_02.py -h
usage: ao2021_s17_02.py [-h] [-x SIZE_X] [-y SIZE_Y] [-n NSTARS] [-f FLUX_MIN]
                        [-p FWHM_PSF] [-d FWHM_STDDEV] [-s SKY]
                        [-e SKY_STDDEV] [-o OUTPUT_FILE]

generating a synthetic image with artificial stars

optional arguments:
  -h, --help            show this help message and exit
  -x SIZE_X, --size-x SIZE_X
                        image size in X-axis (default: 2048
```

```
  -y SIZE_Y, --size-y SIZE_Y
                        image size in Y-axis (default: 2048)
  -n NSTARS, --nstars NSTARS
                        number of stars to generate (default: 100)
  -f FLUX_MIN, --flux-min FLUX_MIN
                        minimum flux of stars (default: 1000)
  -p FWHM_PSF, --fwhm-psf FWHM_PSF
                        FWHM of PSF in pixel (default: 3.5)
  -d FWHM_STDDEV, --fwhm-stddev FWHM_STDDEV
                        stddev of FWHM distribution in pixel (default: 0.1)
  -s SKY, --sky SKY     sky background level in ADU (default: 1000)
  -e SKY_STDDEV, --sky-stddev SKY_STDDEV
                        stddev of sky background in ADU (default: 30)
  -o OUTPUT_FILE, --output-file OUTPUT_FILE
                        output file name

% ./ao2021_s17_02.py -n 200 -o synthetic_0.fits
% ls -l synthetic_0.fits
-rw-r--r--  1 daisuke  taiwan  33560640 May 20 23:06 synthetic_0.fits
```

Use Ginga to display the image. (Fig. 2)

```
% ginga synthetic_0.fits &
```

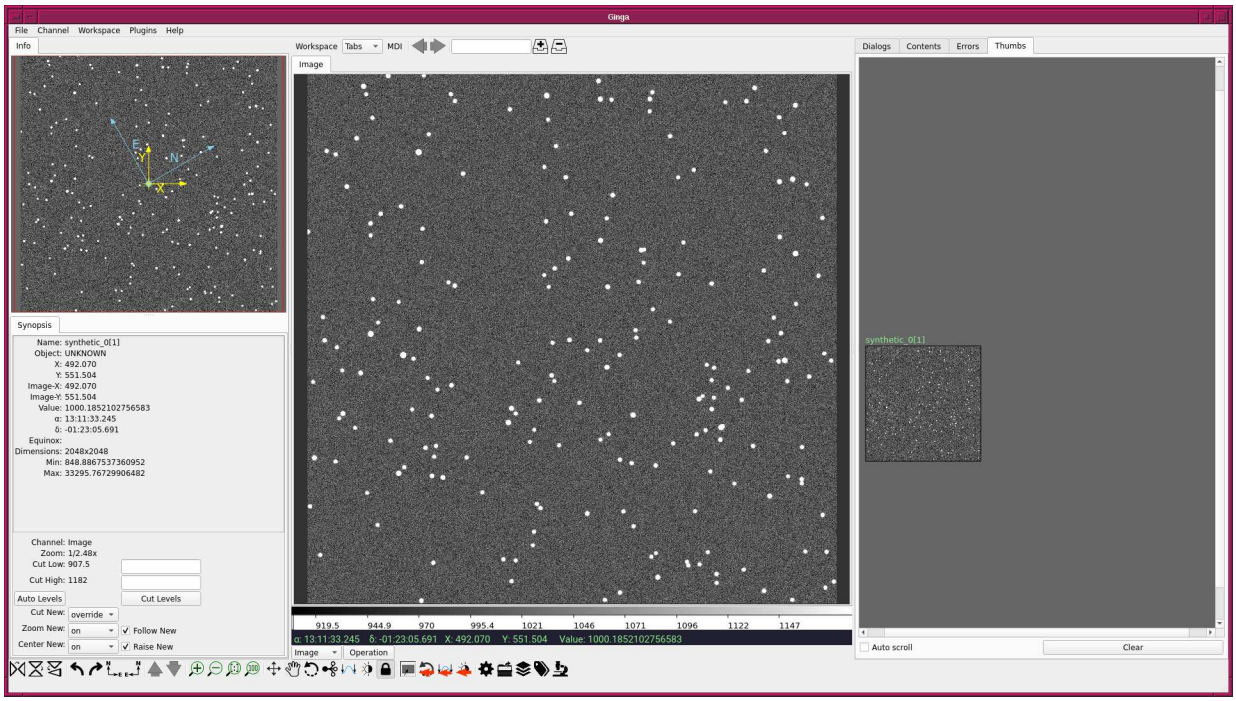

Figure 2: The image of synthetic data generated by `ao2021_s17_02.py`.

2.3 Generating a synthetic image with 200 artificial stars and 30 galaxies

Make a Python script to generate a synthetic image with 200 artificial stars and 30 galaxies.

Python Code 3: ao2021_s17_03.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing numpy module
import numpy
import numpy.random

# importing astropy module
import astropy.table

# importing photutils module
import photutils.datasets

# constructing parser object
desc = 'generating a synthetic image with artificial stars'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-x', '--size-x', type=int, default=2048, \
                     help='image size in X-axis (default: 2048)')
parser.add_argument ('-y', '--size-y', type=int, default=2048, \
                     help='image size in Y-axis (default: 2048)')
parser.add_argument ('-n', '--nstars', type=int, default=100, \
                     help='number of stars to generate (default: 100)')
parser.add_argument ('-g', '--ngalaxies', type=int, default=10, \
                     help='number of galaxies to generate (default: 10)')
parser.add_argument ('-f', '--flux-min', type=float, default=1000.0, \
                     help='minimum flux of stars (default: 1000)')
parser.add_argument ('-p', '--fwhm-psf', type=float, default=3.5, \
                     help='FWHM of PSF in pixel (default: 3.5)')
parser.add_argument ('-d', '--fwhm-stddev', type=float, default=0.1, \
                     help='stddev of FWHM distribution in pixel (default: 0.1)')
parser.add_argument ('-q', '--fwhm-psf-gal', type=float, default=8.0, \
                     help='FWHM of galaxy PSF in pixel (default: 8)')
parser.add_argument ('-r', '--fwhm-psf-gal-stddev', type=float, default=4.0, \
                     help='stddev of FWHM of galaxy PSF in pixel (default: 4)')
parser.add_argument ('-s', '--sky', type=float, default=1000.0, \
                     help='sky background level in ADU (default: 1000)')
parser.add_argument ('-e', '--sky-stddev', type=float, default=30.0, \
                     help='stddev of sky background in ADU (default: 30)')
parser.add_argument ('-o', '--output-file', default='', \
                     help='output file name')

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

# image size
image_size_x = args.size_x
image_size_y = args.size_y
image_shape  = (image_size_x, image_size_y)

# number of stars and galaxies to generate
nstars = args.nstars
```

```
ngals  = args.ngalaxies

# flux of faintest stars
flux_min = args.flux_min

# FWHM of PSF
fwhm_x            = args.fwhm_psf
fwhm_y            = args.fwhm_psf
fwhm_stddev_x     = args.fwhm_stddev
fwhm_stddev_y     = args.fwhm_stddev
fwhm_gal_x        = args.fwhm_psf_gal
fwhm_gal_y        = args.fwhm_psf_gal
fwhm_gal_stddev_x = args.fwhm_psf_gal_stddev
fwhm_gal_stddev_y = args.fwhm_psf_gal_stddev

# sky background level and stddev
sky_mean   = args.sky
sky_stddev = args.sky_stddev

# output file name
file_output = args.output_file

# check of output file name
if not (file_output[-5:] == '.fits'):
    print ("Output file must be a FITS file.")
    sys.exit ()

# generating a new astropy table
table_stars = astropy.table.Table ()
table_gals  = astropy.table.Table ()

# generating random numbers for stars
position_x = numpy.random.default_rng ().uniform (0, image_size_x, nstars)
position_y = numpy.random.default_rng ().uniform (0, image_size_y, nstars)
theta_deg  = numpy.random.default_rng ().uniform (0, 360, nstars)
psf_x      = numpy.random.default_rng ().normal (loc=fwhm_x, \
                                                 scale=fwhm_stddev_x, \
                                                 size=nstars)
psf_y      = numpy.random.default_rng ().normal (loc=fwhm_y, \
                                                 scale=fwhm_stddev_y, \
                                                 size=nstars)
powerlaw   = numpy.random.default_rng ().power (1.5, size=nstars)
flux       = flux_min / powerlaw

# generating random numbers for galaxies
position_gal_x = numpy.random.default_rng ().normal (loc=image_size_x / 2.0, \
                                                     scale=300, size=ngals)
position_gal_y = numpy.random.default_rng ().normal (loc=image_size_y / 2.0, \
                                                     scale=300, size=ngals)
theta_gal_deg  = numpy.random.default_rng ().uniform (0, 360, ngals)
psf_gal_x      = numpy.random.default_rng ().normal (loc=fwhm_gal_x, \
                                                     scale=fwhm_gal_stddev_x, \
                                                     size=ngals)
psf_gal_y      = numpy.random.default_rng ().normal (loc=fwhm_gal_y, \
                                                     scale=fwhm_gal_stddev_y, \
                                                     size=ngals)
powerlaw_gal   = numpy.random.default_rng ().power (2.0, size=ngals)
flux_gal       = flux_min * 3 / powerlaw_gal
```

```
# conversion from degree to radian
theta_rad = numpy.radians (theta_deg)
theta_gal_rad = numpy.radians (theta_gal_deg)

# adding data to the table of stars
table_stars['amplitude'] = flux
table_stars['x_mean']    = position_x
table_stars['y_mean']    = position_y
table_stars['x_stddev']  = psf_x
table_stars['y_stddev']  = psf_y
table_stars['theta']     = theta_rad

# adding data to the table of galaxies
table_gals['amplitude'] = flux_gal
table_gals['x_mean']    = position_gal_x
table_gals['y_mean']    = position_gal_y
table_gals['x_stddev']  = psf_gal_x
table_gals['y_stddev']  = psf_gal_y
table_gals['theta']     = theta_gal_rad

# generating stars
image_stars = photutils.datasets.make_gaussian_sources_image (image_shape, \
                                                               table_stars)

# generating galaxies
image_gals = photutils.datasets.make_gaussian_sources_image (image_shape, \
                                                             table_gals)

# generating sky background
image_sky = photutils.datasets.make_noise_image (image_shape, \
                                                 distribution='gaussian', \
                                                 mean=sky_mean, \
                                                 stddev=sky_stddev)
# generating synthetic image
image = image_stars + image_gals + image_sky

# making WCS
wcs = photutils.datasets.make_wcs (image_shape)

# making Image HDU
hdu = photutils.datasets.make_imagehdu (image, wcs=wcs)

# writing a FITS file
hdu.writeto (file_output)
```

Execute the script, and generate a FITS file.

```
% chmod a+x ao2021_s17_03.py
% ./ao2021_s17_03.py -h
usage: ao2021_s17_03.py [-h] [-x SIZE_X] [-y SIZE_Y] [-n NSTARS]
                        [-g NGALAXIES] [-f FLUX_MIN] [-p FWHM_PSF]
                        [-d FWHM_STDDEV] [-q FWHM_PSF_GAL]
                        [-r FWHM_PSF_GAL_STDDEV] [-s SKY] [-e SKY_STDDEV]
                        [-o OUTPUT_FILE]

generating a synthetic image with artificial stars

optional arguments:
```

```
  -h, --help            show this help message and exit
  -x SIZE_X, --size-x SIZE_X
                        image size in X-axis (default: 2048
  -y SIZE_Y, --size-y SIZE_Y
                        image size in Y-axis (default: 2048)
  -n NSTARS, --nstars NSTARS
                        number of stars to generate (default: 100)
  -g NGALAXIES, --ngalaxies NGALAXIES
                        number of galaxies to generate (default: 10)
  -f FLUX_MIN, --flux-min FLUX_MIN
                        minimum flux of stars (default: 1000)
  -p FWHM_PSF, --fwhm-psf FWHM_PSF
                        FWHM of PSF in pixel (default: 3.5)
  -d FWHM_STDDEV, --fwhm-stddev FWHM_STDDEV
                        stddev of FWHM distribution in pixel (default: 0.1)
  -q FWHM_PSF_GAL, --fwhm-psf-gal FWHM_PSF_GAL
                        FWHM of galaxy PSF in pixel (default: 8)
  -r FWHM_PSF_GAL_STDDEV, --fwhm-psf-gal-stddev FWHM_PSF_GAL_STDDEV
                        stddev of FWHM of galaxy PSF in pixel (default: 4)
  -s SKY, --sky SKY     sky background level in ADU (default: 1000)
  -e SKY_STDDEV, --sky-stddev SKY_STDDEV
                        stddev of sky background in ADU (default: 30)
  -o OUTPUT_FILE, --output-file OUTPUT_FILE
                        output file name

% ./ao2021_s17_03.py -n 200 -g 30 -o synthetic_1.fits
% ls -l synthetic_1.fits
-rw-r--r--  1 daisuke  taiwan  33560640 May 20 23:16 synthetic_1.fits
```

Use Ginga to display the image. (Fig. 3)

```
% ginga synthetic_1.fits &
```

3 Detecting and masking sources

Make a Python script to detect and mask sources on the image.

Python Code 4: ao2021_s17_04.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing datetime module
import datetime

# importing numpy module
import numpy.ma

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

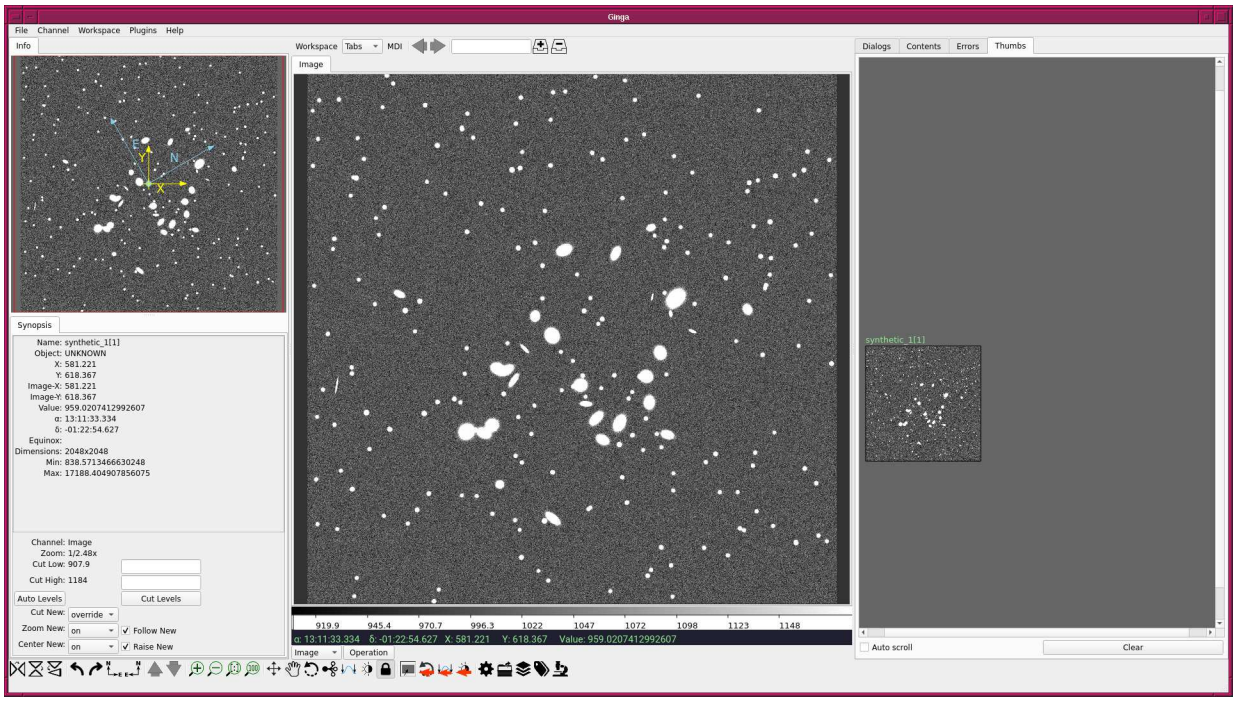

Figure 3: A synthetic image of 200 stars and 30 galaxies.

```
# importing photutils module
import photutils.segmentation

# constructing parser object
desc = 'detecting and masking sources'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-i', '--input-file', default='', \
                     help='input file name')
parser.add_argument ('-o', '--output-file', default='', \
                     help='output file name')
parser.add_argument ('-t', '--threshold', type=float, default=2.0, \
                     help='detection threshold in sigma (default: 2)')
parser.add_argument ('-n', '--npixels', type=int, default=5, \
                     help='minimum number of pixels for detection (default: 5)')
parser.add_argument ('-s', '--dilate-size', type=int, default=21, \
                     help='dilate size (default: 21)')
parser.add_argument ('-m', '--maxiters', type=int, default=30, \
                     help='maximum number of iterations (default: 30)')

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

# file names
file_input  = args.input_file
file_output = args.output_file

# input parameters
threshold   = args.threshold
npixels     = args.npixels
dilate_size = args.dilate_size
maxiters    = args.maxiters
```

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

# check of output file name
if not (file_output[-5:] == '.fits'):
    print ("Output file must be a FITS file.")
    sys.exit ()

# opening FITS file
with astropy.io.fits.open (file_input) as hdu:
    # reading header and image
    header = hdu[0].header
    image  = hdu[0].data
    # if no image in PrimaryHDU, then read next HDU
    if (header['NAXIS'] == 0):
        header = hdu[1].header
        image  = hdu[1].data

# making source mask
source_mask \
    = photutils.segmentation.make_source_mask (image, threshold,
                                               npixels=npixels,
                                               sigclip_iters=maxiters,
                                               dilate_size=dilate_size)

# making masked array
image_masked = numpy.ma.array (image, mask=source_mask)

# now
datetime_now = datetime.datetime.now ()

# adding comments in header
header['comment'] = "Updated on %s" % (datetime_now)
header['comment'] = "Detecting and masking sources"
header['comment'] = "Original file = %s" % (file_input)
header['comment'] = "Options:"
header['comment'] = "  threshold   = %f sigma" % (threshold)
header['comment'] = "  npixels     = %d pixels" % (npixels)
header['comment'] = "  dilate_size = %d pixels" % (dilate_size)
header['comment'] = "  maxiters    = %d" % (maxiters)

# writing a FITS file
astropy.io.fits.writeto (file_output, \
                         numpy.ma.filled (image_masked, fill_value=0.0), \
                         header=header)
```

Run the script.

```
% chmod a+x ao2021_s17_04.py
% ./ao2021_s17_04.py -h
usage: ao2021_s17_04.py [-h] [-i INPUT_FILE] [-o OUTPUT_FILE] [-t THRESHOLD]
                        [-n NPIXELS] [-s DILATE_SIZE] [-m MAXITERS]

detecting and masking sources
```

```
optional arguments:
  -h, --help            show this help message and exit
  -i INPUT_FILE, --input-file INPUT_FILE
                        input file name
  -o OUTPUT_FILE, --output-file OUTPUT_FILE
                        output file name
  -t THRESHOLD, --threshold THRESHOLD
                        detection threshold in sigma (default: 2)
  -n NPIXELS, --npixels NPIXELS
                        minimum number of pixels for detection (default: 5)
  -s DILATE_SIZE, --dilate-size DILATE_SIZE
                        dilate size (default: 21)
  -m MAXITERS, --maxiters MAXITERS
                        maximum number of iterations (default: 30)

% ./ao2021_s17_04.py -i synthetic_1.fits -o synthetic_1_masked.fits
% ls -l synthetic_1*
-rw-r--r--  1 daisuke  taiwan  33560640 May 20 23:16 synthetic_1.fits
-rw-r--r--  1 daisuke  taiwan  33557760 May 21 00:36 synthetic_1_masked.fits
```

Use Ginga to display the image. (Fig. 4)

```
% ginga synthetic_1_masked.fits &
```

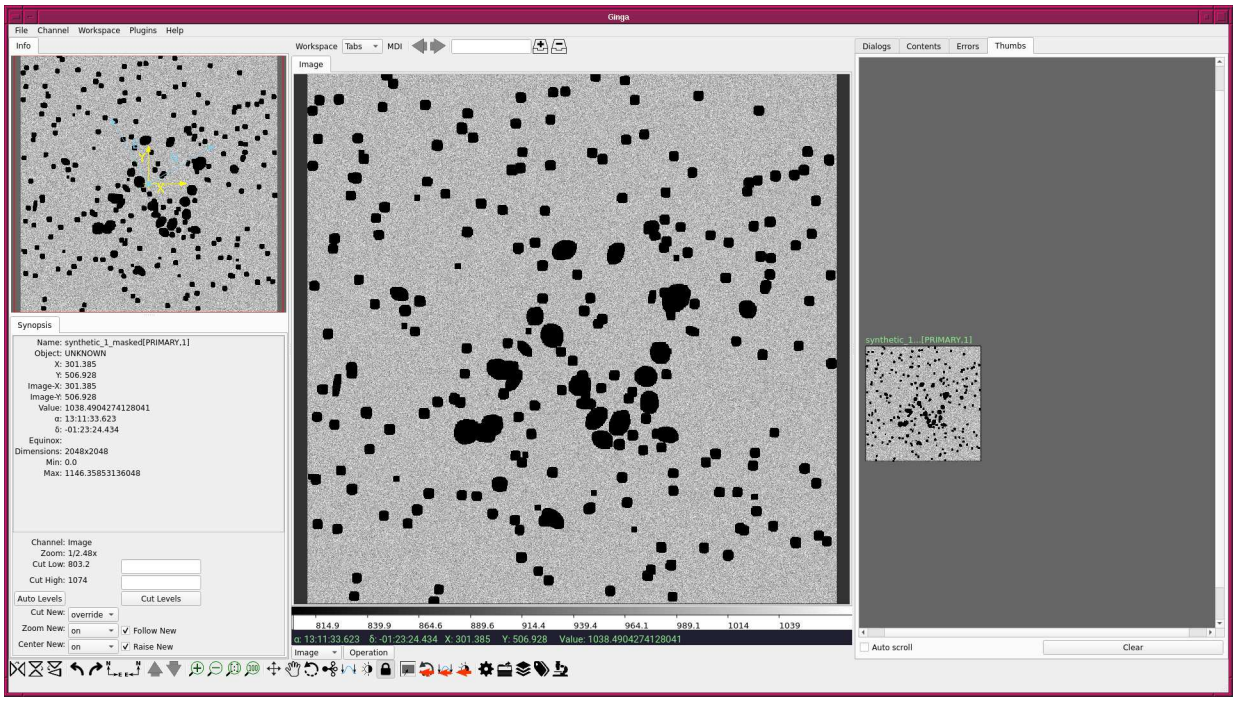

Figure 4: Masked image. Stars and galaxies are successfully masked.

4 Background estimation

Make a Python script to estimate background level.

Python Code 5: ao2021_s17_05.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing datetime module
import datetime

# importing numpy module
import numpy.ma

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

# importing photutils module
import photutils.segmentation

# constructing parser object
desc = 'detecting and masking sources'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-i', '--input-file', default='', \
                     help='input file name')
parser.add_argument ('-t', '--threshold', type=float, default=2.0, \
                     help='detection threshold in sigma (default: 2)')
parser.add_argument ('-n', '--npixels', type=int, default=5, \
                     help='minimum number of pixels for detection (default: 5)')
parser.add_argument ('-s', '--dilate-size', type=int, default=21, \
                     help='dilate size (default: 21)')
parser.add_argument ('-m', '--maxiters', type=int, default=30, \
                     help='maximum number of iterations (default: 30)')
parser.add_argument ('-r', '--sigma-clipping', type=float, default=4.0, \
                     help='sigma-clipping threshold in sigma (default: 4)')

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

# file names
file_input  = args.input_file

# input parameters
threshold   = args.threshold
npixels     = args.npixels
dilate_size = args.dilate_size
maxiters    = args.maxiters
rejection   = args.sigma_clipping

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

# opening FITS file
```

```
with astropy.io.fits.open (file_input) as hdu:
    # reading header and image
    header = hdu[0].header
    image  = hdu[0].data
    # if no image in PrimaryHDU, then read next HDU
    if (header['NAXIS'] == 0):
        header = hdu[1].header
        image  = hdu[1].data

# making source mask
source_mask \
    = photutils.segmentation.make_source_mask (image, threshold,
                                               npixels=npixels,
                                               sigclip_iters=maxiters,
                                               dilate_size=dilate_size)

# making masked array
image_masked = numpy.ma.array (image, mask=source_mask)

# sigma-clipping
skybg_mean, skybg_median, skybg_stddev \
    = astropy.stats.sigma_clipped_stats (image, sigma=rejection)

# mode calculation using empirical formula
skybg_mode = 3.0 * skybg_median - 2.0 * skybg_mean

# printing results
print ("mean   = %f" % skybg_mean)
print ("median = %f" % skybg_median)
print ("mode   = %f" % skybg_mode)
print ("stddev = %f" % skybg_stddev)
```

Execute the script.

```
% chmod a+x ao2021_s17_05.py
% ./ao2021_s17_05.py -h
usage: ao2021_s17_05.py [-h] [-i INPUT_FILE] [-t THRESHOLD] [-n NPIXELS]
                        [-s DILATE_SIZE] [-m MAXITERS] [-r SIGMA_CLIPPING]

detecting and masking sources

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT_FILE, --input-file INPUT_FILE
                        input file name
  -t THRESHOLD, --threshold THRESHOLD
                        detection threshold in sigma (default: 2)
  -n NPIXELS, --npixels NPIXELS
                        minimum number of pixels for detection (default: 5)
  -s DILATE_SIZE, --dilate-size DILATE_SIZE
                        dilate size (default: 21)
  -m MAXITERS, --maxiters MAXITERS
                        maximum number of iterations (default: 30)
  -r SIGMA_CLIPPING, --sigma-clipping SIGMA_CLIPPING
                        sigma-clipping threshold in sigma (default: 4)

% ./ao2021_s17_05.py -i synthetic_0.fits
mean   = 1000.412907
```

```
median = 1000.277723
mode   = 1000.007355
stddev = 30.382858
% ./ao2021_s17_05.py -i synthetic_1.fits
mean   = 1000.746540
median = 1000.476765
mode   = 999.937214
stddev = 30.671132
```

The mode seems to be a good estimator of the sky background level.

5 Dealing with real data

Copy a reduced image, and check the mask. (Fig. 5)

```
% cp -pi ../../16_skybackground/script_16/lot_20210214_0185_df.fits .
% ls -l lot_20210214_0185_df.fits
-rw-r--r--  1 daisuke  taiwan  33566400 May 11 12:38 lot_20210214_0185_df.fits
% ./ao2021_s17_04.py -i lot_20210214_0185_df.fits -o 0185_masked.fits
% ls -l 0185_masked.fits
-rw-r--r--  1 daisuke  taiwan  33569280 May 21 00:58 0185_masked.fits
% ginga 0185_masked.fits &
```

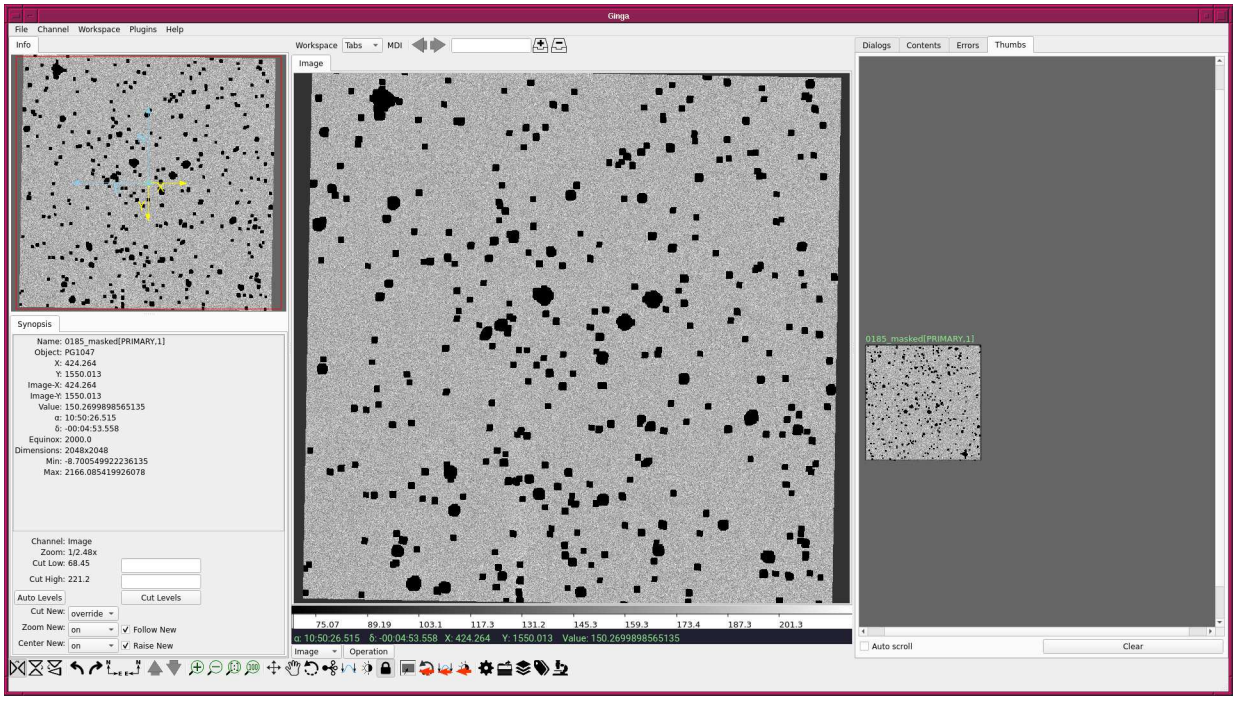

Figure 5: Masked image of `lot_20210214_0185_df.fits`.

Estimate the sky background level.

```
% ./ao2021_s17_05.py -i lot_20210214_0185_df.fits
mean   = 175.566294
median = 175.347725
mode   = 174.910588
stddev = 16.413741
```

6 For your training

Read followings.

1. “Data Tables” of Astropy
   - `https://docs.astropy.org/en/stable/table/index.html`
2. “FITS File Handling” of Astropy
   - `https://docs.astropy.org/en/stable/io/fits/index.html`
3. “Random sampling” of Numpy
   - `https://numpy.org/doc/stable/reference/random/index.html`
4. “Datasets” of photutils
   - `https://photutils.readthedocs.io/en/stable/datasets.html`
5. “Image Segmentation”
   - `https://photutils.readthedocs.io/en/stable/segmentation.html`

7 Assignment

None.