

Astroinformatics 2022

Session 10: Distribution of asteroids, stars, and galaxies

Kinoshita Daisuke

21 November 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 “Astroinformatics” (course ID: AS6095) offered at Institute of Astronomy, National Central University from September 2022 to January 2023.
 - The file is provided in the hope that it will be useful, but there is no guarantee for the correctness. Use this file at your own risk.
 - If you are willing to use this file for your study, please feel free to use. I’ll be very happy to receive feedback from you.
 - If you are willing to use this file for your teaching, please contact to Kinoshita Daisuke. When you use this file partly or entirely, please mention clearly that the author of the original version is Kinoshita Daisuke. Please help me to improve the contents of this file by sending your feedback.
 - Contact address: <https://www.instagram.com/daisuke23888/>

For this session, we make plots for distribution of asteroids, stars, and galaxies.

1 Sample Python scripts for this session

Sample Python scripts for this session can be downloaded from GitHub repository. Visit following GitHub repository.

- https://github.com/kinoshitadaisuke/ncu_astroinformatics_202209

1.1 Executing sample Python scripts on a terminal emulator

If you prefer to execute sample Python scripts for this session on a terminal emulator, download `.py` files from GitHub repository.

1.2 Executing sample Python scripts on JupyterLab

If you prefer to execute sample Python scripts for this session on JupyterLab (or Jupyter Notebook), download `.ipynb` file from GitHub repository.

1.3 Executing sample Python scripts using Binder

If you prefer to execute sample Python scripts for this session on Binder, visit following web page.

- https://mybinder.org/v2/gh/kinoshitadaisuke/ncu_astroinformatics_202209/HEAD

Start your favourite web browser and go to above web page. (Fig. 1) In a minute or two, you see JupyterLab working on your web browser. (Fig. 2) Go to the directory (folder) “s10”. (Fig. 3) Choose the file “ai202209_s10.ipynb” (Fig. 4 and 5) and open it (Fig. 6).

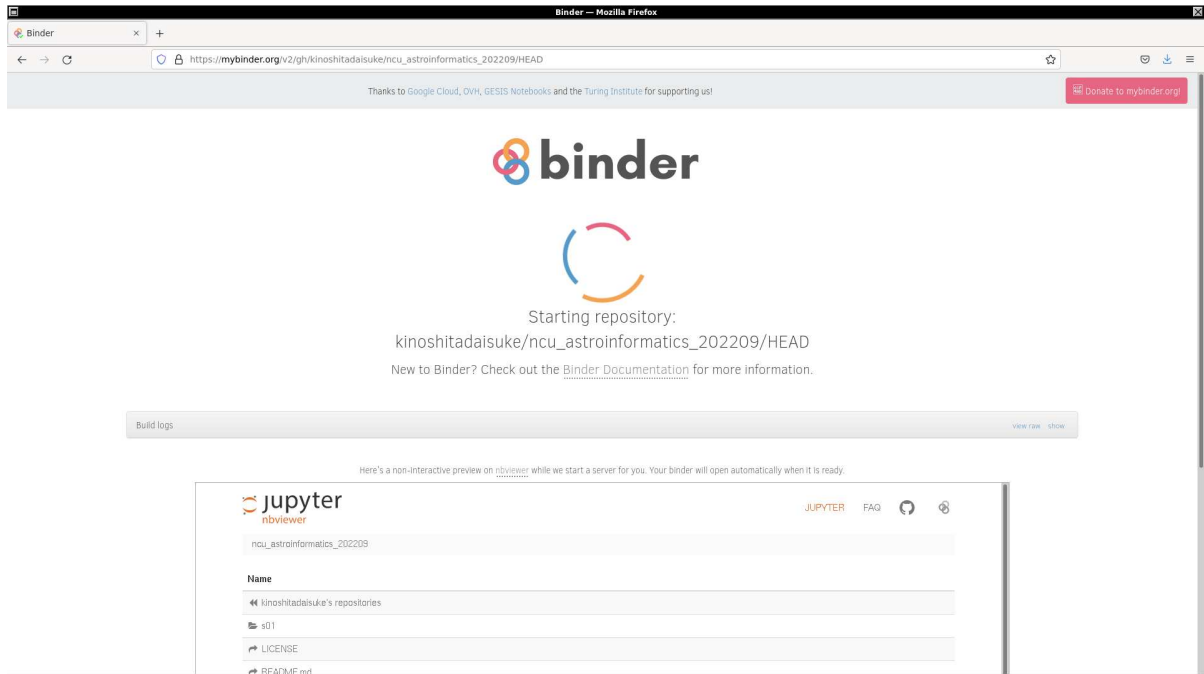


Figure 1: Using Binder to execute sample Python scripts for this session.

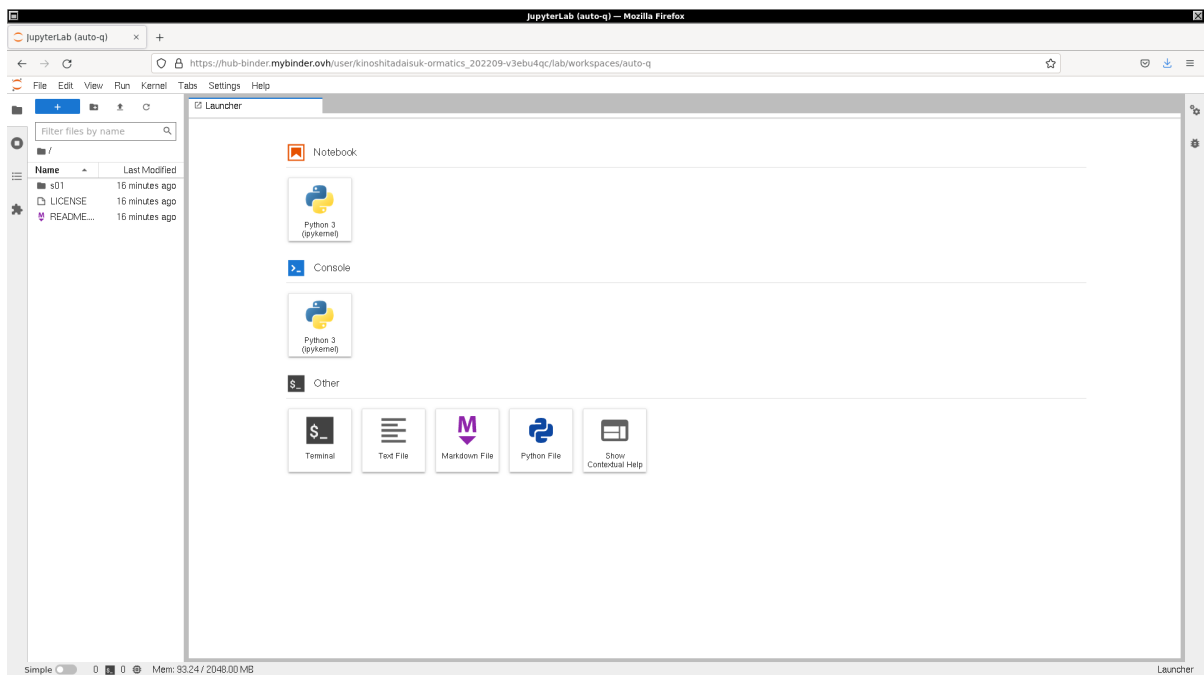


Figure 2: Using Binder to execute sample Python scripts for this session.

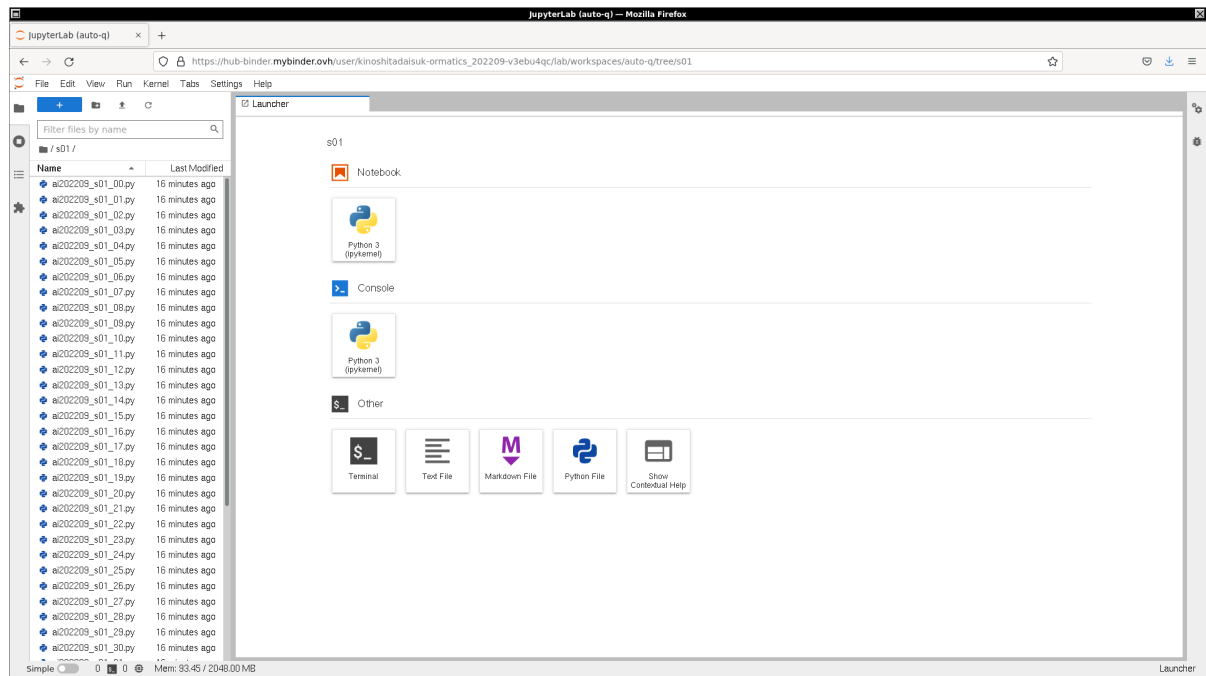


Figure 3: Using Binder to execute sample Python scripts for this session.

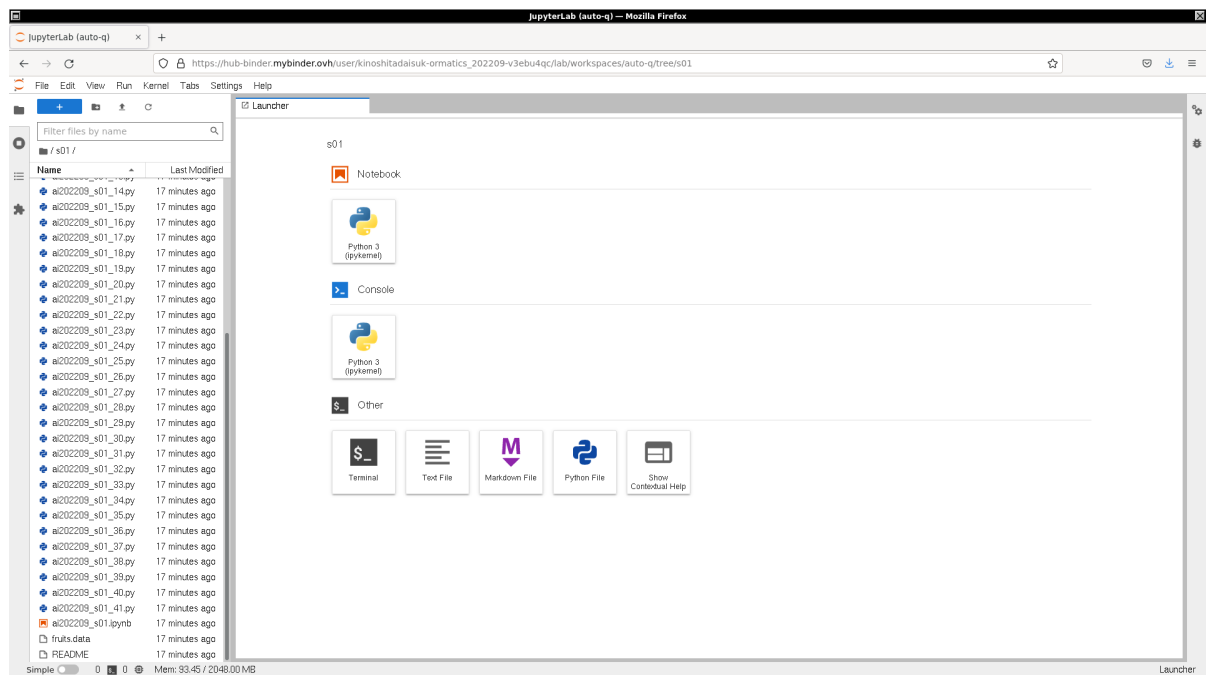


Figure 4: Using Binder to execute sample Python scripts for this session.

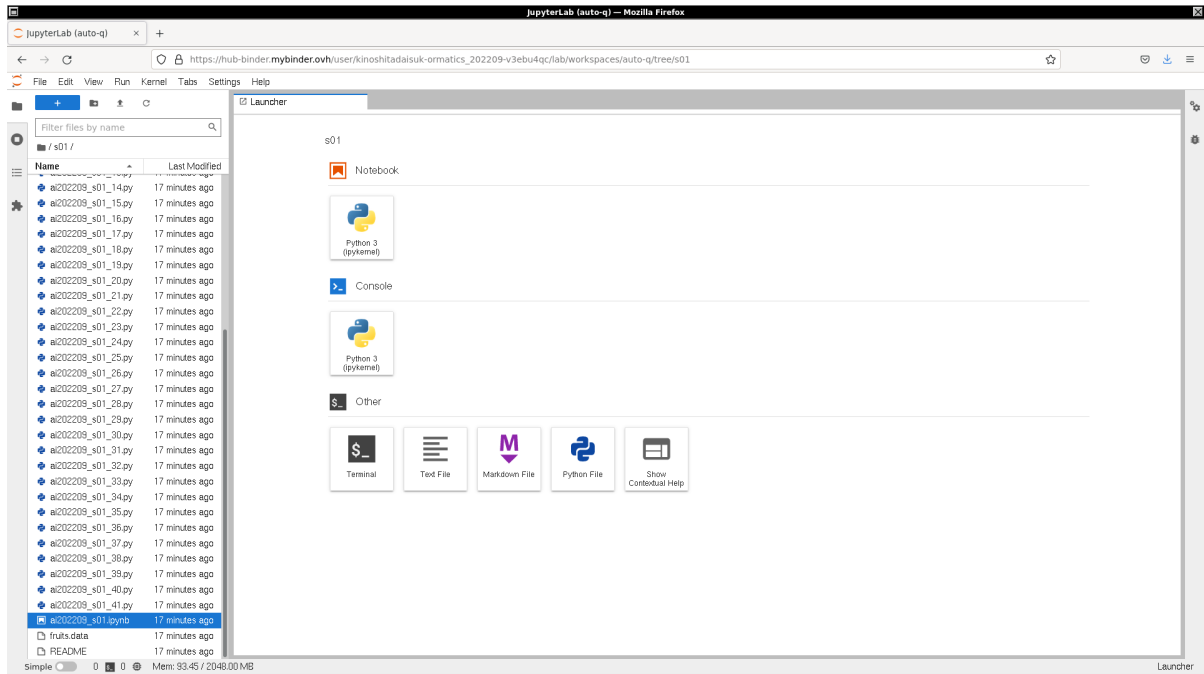


Figure 5: Using Binder to execute sample Python scripts for this session.

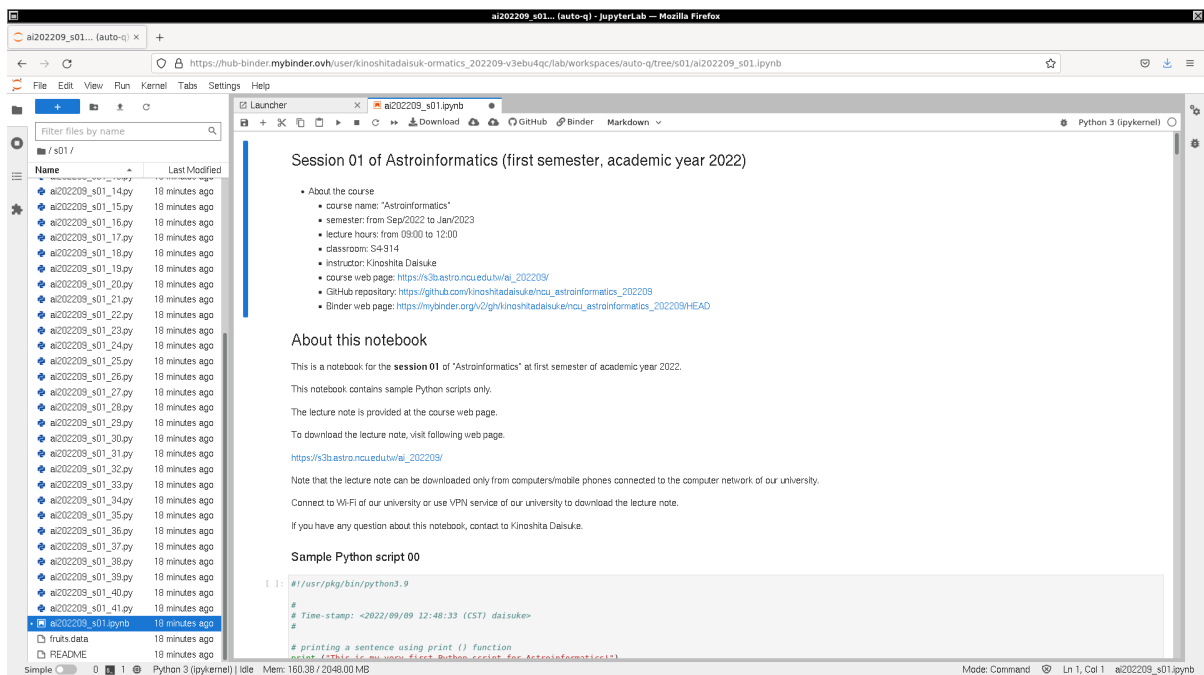


Figure 6: Using Binder to execute sample Python scripts for this session.

2 Distribution of asteroids

2.1 Downloading orbital elements of known asteroids

Make a Python script to check whether or not you have the asteroid orbit database file “MPCORB.DAT”.

Python Code 1: ai202209_s10_00_00.py

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

#
# Time-stamp: <2022/11/18 11:08:41 (CST) daisuke>
#

# importing pathlib module
import pathlib

# file names
list_filename = ['MPCORB.DAT', 'MPCORB.DAT.gz', \
                 'mpcorb.dat', 'mpcorb.data', \
                 'mpcorb.dat.gz', 'mpcorb.data.gz']

# existence check of files
for filename in list_filename:
    # making pathlib object
    path_mpcorb = pathlib.Path (filename)
    # existence check
    existence = path_mpcorb.exists ()
    # printing result of existence check
    if (existence):
        print (f"{filename:24s} : exists")
    else:
        print (f"{filename:24s} : DOES NOT exist")
```

Execute above script to check the existence of file.

```
% chmod a+x ai202209_s10_00_00.py
% ./ai202209_s10_00_00.py
MPCORB.DAT           : DOES NOT exist
MPCORB.DAT.gz        : DOES NOT exist
mpcorb.dat           : DOES NOT exist
mpcorb.data          : DOES NOT exist
mpcorb.dat.gz        : DOES NOT exist
mpcorb.data.gz       : DOES NOT exist
```

Try following practice.

Practice 10-01

Make a Python script to check the existence of the file “/etc/fstab”.

If you do not have the file “MPCORB.DAT”, download it from the Minor Planet Center of IAU. Here is a sample script for downloading the file “MPCORB.DAT”.

Python Code 2: ai202209_s10_00_01.py

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

#
# Time-stamp: <2022/11/18 11:31:24 (CST) daisuke>
#
```

```

# importing urllib module
import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# URL of data file
url_data = 'https://www.minorplanetcenter.net/iau/MPCORB/MPCORB.DAT.gz'

# output file name
file_output = 'mpcorb.dat.gz'

# printing status
print (f'Now, fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()

# printing status
print (f'Finished fetching {url_data}!')

# printing status
print (f'Now, writing the data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'wb') as fh_write:
    # writing data
    fh_write.write (data_byte)

# printing status
print (f'Finished writing the data into file "{file_output}"!')

```

Execute above script to check the existence of file.

```

% chmod a+x ai202209_s10_00_01.py
% ./ai202209_s10_00_01.py
Now, fetching https://www.minorplanetcenter.net/iau/MPCORB/MPCORB.DAT.gz...
Finished fetching https://www.minorplanetcenter.net/iau/MPCORB/MPCORB.DAT.gz!
Now, writing the data into file "mpcorb.dat.gz"...
Finished writing the data into file "mpcorb.dat.gz"!
% ls -lF mpcorb.dat.gz
-rw-r--r--  1 daisuke  wheel  74756295 Nov 18 11:20 mpcorb.dat.gz
% file mpcorb.dat.gz
mpcorb.dat.gz: gzip compressed data, was "MPCORB.DAT", last modified: Thu Nov 17
12:40:37 2022, max compression, from Unix, original size modulo 2^32 254308368

```

Try following practice.

Practice 10-02

Make a Python script to download the file “<http://tdc-www.harvard.edu/catalogs/bsc5.dat.gz>”.

2.2 Reading asteroid orbit database file

Make a Python script to read MPC's asteroid orbit database file. To learn about the format of the file "MPCORB.DAT", read following description.

- Export Format for Minor-Planet Orbits: <https://www.minorplanetcenter.net/iau/info/MPOrbitFormat.html>

Here is a sample script to read the file "MPCORB.DAT".

Python Code 3: ai202209_s10_00_02.py

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

#
# Time-stamp: <2022/11/18 11:33:12 (CST) daisuke>
#

# importing gzip module
import gzip

# catalogue file name
file_catalogue = 'mpcorb.dat.gz'

# counter
counter = 0

# opening catalogue file
with gzip.open (file_catalogue, 'rb') as fh:
    # reading catalogue line-by-line
    for line in fh:
        # number of provisional designation
        try:
            designation = line[0:7].strip ().decode ('utf-8')
        except:
            continue
        # absolute magnitude
        try:
            absmag = float (line[8:13])
        except:
            absmag = -999.99
        # mean anomaly
        try:
            M = float (line[26:35])
        except:
            M = -999.99
        # argument of perihelion
        try:
            peri = float (line[37:46])
        except:
            peri = -999.99
        # longitude of ascending node
        try:
            node = float (line[48:57])
        except:
            node = -999.99
        # inclination
        try:
            i = float (line[59:68])
        except:
            i = -999.99
        # eccentricity
```

```

try:
    e = float (line[70:79])
except:
    e = -999.99
# semimajor axis
try:
    a = float (line[92:103])
except:
    a = -999.99
# number of observations
try:
    nobs = int (line[117:122])
except:
    nobs = -999
# residual
try:
    residual = float (line[137:141])
except:
    residual = -999.99
# 4-hexdigit flags
try:
    flag = line[161:165].strip ().decode ('utf-8')
except:
    flag = '9999'
# readable name
try:
    name = line[166:194].strip ().decode ('utf-8')
except:
    name = '__NONE__'
# last observation date
try:
    lastobs = int (line[194:202])
except:
    lastobs = 99999999

# skip when reading the header
if ( (a < -999.0) and (e < -999.0) and (i < -999.0) \
    and (peri < -999.0) and (node < -999.0) and (M < -999.0) ):
    continue

# printing orbital elements of first 5 asteroids
if (counter < 5):
    print (f'designation = {designation}')
    print (f'  name      = {name}')
    print (f'  absmag    = {absmag}')
    print (f'  M        = {M}')
    print (f'  peri      = {peri}')
    print (f'  node      = {node}')
    print (f'  i        = {i}')
    print (f'  e        = {e}')
    print (f'  a        = {a}')
    print (f'  nobs      = {nobs}')
    print (f'  residual = {residual}')
    print (f'  flag       = {flag}')
    print (f'  lastobs   = {lastobs}')

# increment counter
counter += 1

```

Execute above script to read the file "MPCORB.DAT".

```
% chmod a+x ai202209_s10_00_02.py
% ./ai202209_s10_00_02.py
designation = 00001
  name      = (1) Ceres
  absmag    = 3.33
  M         = 334.32722
  peri      = 73.53161
  node      = 80.26642
  i         = 10.5868
  e         = 0.0786358
  a         = 2.766619
  nobs      = 7258
  residual  = 0.65
  flag      = 0000
  lastobs   = 20220916
designation = 00002
  name      = (2) Pallas
  absmag    = 4.12
  M         = 315.09111
  peri      = 310.84262
  node      = 172.91791
  i         = 34.92715
  e         = 0.2299864
  a         = 2.7694633
  nobs      = 8871
  residual  = 0.59
  flag      = 0000
  lastobs   = 20220804
designation = 00003
  name      = (3) Juno
  absmag    = 5.14
  M         = 306.62242
  peri      = 247.80396
  node      = 169.84591
  i         = 12.99223
  e         = 0.256775
  a         = 2.6704222
  nobs      = 7414
  residual  = 0.63
  flag      = 0000
  lastobs   = 20221111
designation = 00004
  name      = (4) Vesta
  absmag    = 3.21
  M         = 61.19232
  peri      = 151.25772
  node      = 103.80084
  i         = 7.14078
  e         = 0.0884019
  a         = 2.3619872
  nobs      = 7522
  residual  = 0.63
  flag      = 0000
  lastobs   = 20221111
designation = 00005
  name      = (5) Astraea
  absmag    = 7.01
```

```
M          = 208.56769
peri       = 358.7923
node       = 141.56759
i          = 5.36542
e          = 0.1889852
a          = 2.5775402
nobs       = 3126
residual   = 0.72
flag       = 0000
lastobs    = 20221113
```

Try following practice.

Practice 10-03

Make a Python script to read the Bright Star Catalogue “bsc5.dat.gz”. To learn about the format of the catalogue file, read the document <http://tdc-www.harvard.edu/catalogs/bsc5.readme>.

2.3 Semimajor axis distribution of asteroids

Construct a histogram of semimajor axis of asteroids. Here is a sample script.

Python Code 4: ai202209_s10_00_03.py

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

#
# Time-stamp: <2022/11/18 22:16:28 (CST) daisuke>
#

# importing gzip module
import gzip

# importing numpy module
import numpy

# catalogue file name
file_catalogue = 'mpcorb.dat.gz'

#
# parameters for histogram
#

# width of a bin in au
bin_width = 0.1

# number of bins
n_bins = 100

# making a numpy array for histogram
hist_a = numpy.zeros (n_bins, dtype=int)

# opening catalogue file
with gzip.open (file_catalogue, 'rb') as fh:
    # reading catalogue line-by-line
    for line in fh:
        # number of provisional designation
        try:
            designation = line[0:7].strip ().decode ('utf-8')
```

```
except:
    continue
# absolute magnitude
try:
    absmag = float (line[8:13])
except:
    absmag = -999.99
# mean anomaly
try:
    M = float (line[26:35])
except:
    M = -999.99
# argument of perihelion
try:
    peri = float (line[37:46])
except:
    peri = -999.99
# longitude of ascending node
try:
    node = float (line[48:57])
except:
    node = -999.99
# inclination
try:
    incl = float (line[59:68])
except:
    incl = -999.99
# eccentricity
try:
    e = float (line[70:79])
except:
    e = -999.99
# semimajor axis
try:
    a = float (line[92:103])
except:
    a = -999.99
# number of observations
try:
    nobs = int (line[117:122])
except:
    nobs = -999
# residual
try:
    residual = float (line[137:141])
except:
    residual = -999.99
# 4-hexdigit flags
try:
    flag = line[161:165].strip ().decode ('utf-8')
except:
    flag = '9999'
# readable name
try:
    name = line[166:194].strip ().decode ('utf-8')
except:
    name = '__NONE__'
# last observation date
try:
```

```

        lastobs = int (line[194:202])
    except:
        lastobs = 99999999

    # skip when reading the header
    if ( (a < -999.0) and (e < -999.0) and (incl < -999.0) \
        and (peri < -999.0) and (node < -999.0) and (M < -999.0) ):
        continue

    # counting number of asteroids in each bin
    if (a * 10 < n_bins):
        i = int (a * 10)
        hist_a[i] += 1

# printing histogram
for i in range (n_bins):
    start = i * bin_width
    end   = (i + 1) * bin_width
    print (f"from {start:5.2f} au to {end:5.2f} au : {hist_a[i]:8d}")

```

Execute above script to read the file “MPCORB.DAT” and construct a histogram of semimajor axis.

```

% chmod a+x ai202209_s10_00_03.py
% ./ai202209_s10_00_03.py
from 0.00 au to 0.10 au :      0
from 0.10 au to 0.20 au :      0
from 0.20 au to 0.30 au :      0
from 0.30 au to 0.40 au :      0
from 0.40 au to 0.50 au :       1
from 0.50 au to 0.60 au :       5
from 0.60 au to 0.70 au :      58
from 0.70 au to 0.80 au :     308
from 0.80 au to 0.90 au :     823
from 0.90 au to 1.00 au :    1222
from 1.00 au to 1.10 au :    1551
from 1.10 au to 1.20 au :    1757
from 1.20 au to 1.30 au :    1942
from 1.30 au to 1.40 au :    2117
from 1.40 au to 1.50 au :    2013
from 1.50 au to 1.60 au :    1898
from 1.60 au to 1.70 au :    2070
from 1.70 au to 1.80 au :    2729
from 1.80 au to 1.90 au :   14600
from 1.90 au to 2.00 au :   19592
from 2.00 au to 2.10 au :    2599
from 2.10 au to 2.20 au :   25345
from 2.20 au to 2.30 au :   94230
from 2.30 au to 2.40 au :  145034
from 2.40 au to 2.50 au :   77760
from 2.50 au to 2.60 au :  123998
from 2.60 au to 2.70 au :  168281
from 2.70 au to 2.80 au :  133976
from 2.80 au to 2.90 au :   45824
from 2.90 au to 3.00 au :   53400
from 3.00 au to 3.10 au :  123214
from 3.10 au to 3.20 au :  150576
from 3.20 au to 3.30 au :   26617
from 3.30 au to 3.40 au :    2475
from 3.40 au to 3.50 au :    2298

```

from	3.50	au	to	3.60	au	:	624
from	3.60	au	to	3.70	au	:	134
from	3.70	au	to	3.80	au	:	81
from	3.80	au	to	3.90	au	:	169
from	3.90	au	to	4.00	au	:	5537
from	4.00	au	to	4.10	au	:	437
from	4.10	au	to	4.20	au	:	45
from	4.20	au	to	4.30	au	:	44
from	4.30	au	to	4.40	au	:	45
from	4.40	au	to	4.50	au	:	33
from	4.50	au	to	4.60	au	:	32
from	4.60	au	to	4.70	au	:	15
from	4.70	au	to	4.80	au	:	20
from	4.80	au	to	4.90	au	:	30
from	4.90	au	to	5.00	au	:	62
from	5.00	au	to	5.10	au	:	334
from	5.10	au	to	5.20	au	:	5387
from	5.20	au	to	5.30	au	:	5951
from	5.30	au	to	5.40	au	:	553
from	5.40	au	to	5.50	au	:	21
from	5.50	au	to	5.60	au	:	14
from	5.60	au	to	5.70	au	:	18
from	5.70	au	to	5.80	au	:	20
from	5.80	au	to	5.90	au	:	10
from	5.90	au	to	6.00	au	:	13
from	6.00	au	to	6.10	au	:	8
from	6.10	au	to	6.20	au	:	10
from	6.20	au	to	6.30	au	:	12
from	6.30	au	to	6.40	au	:	9
from	6.40	au	to	6.50	au	:	7
from	6.50	au	to	6.60	au	:	6
from	6.60	au	to	6.70	au	:	6
from	6.70	au	to	6.80	au	:	5
from	6.80	au	to	6.90	au	:	6
from	6.90	au	to	7.00	au	:	7
from	7.00	au	to	7.10	au	:	6
from	7.10	au	to	7.20	au	:	4
from	7.20	au	to	7.30	au	:	7
from	7.30	au	to	7.40	au	:	3
from	7.40	au	to	7.50	au	:	3
from	7.50	au	to	7.60	au	:	5
from	7.60	au	to	7.70	au	:	10
from	7.70	au	to	7.80	au	:	2
from	7.80	au	to	7.90	au	:	4
from	7.90	au	to	8.00	au	:	4
from	8.00	au	to	8.10	au	:	2
from	8.10	au	to	8.20	au	:	5
from	8.20	au	to	8.30	au	:	5
from	8.30	au	to	8.40	au	:	5
from	8.40	au	to	8.50	au	:	5
from	8.50	au	to	8.60	au	:	3
from	8.60	au	to	8.70	au	:	1
from	8.70	au	to	8.80	au	:	5
from	8.80	au	to	8.90	au	:	3
from	8.90	au	to	9.00	au	:	6
from	9.00	au	to	9.10	au	:	4
from	9.10	au	to	9.20	au	:	3
from	9.20	au	to	9.30	au	:	4
from	9.30	au	to	9.40	au	:	4

```

from 9.40 au to 9.50 au :      1
from 9.50 au to 9.60 au :      2
from 9.60 au to 9.70 au :      6
from 9.70 au to 9.80 au :      5
from 9.80 au to 9.90 au :      3
from 9.90 au to 10.00 au :     3

```

Try following practice.

Practice 10-04

Read the file “MPCORB.DAT” and construct a histogram of eccentricity of asteroids.

Construct a histogram of semimajor axis of asteroids and make a plot. Here is a sample script.

Python Code 5: ai202209_s10_00_04.py

```

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

#
# Time-stamp: <2022/11/18 23:00:20 (CST) daisuke>
#

# importing gzip module
import gzip

# importing numpy module
import numpy

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

# catalogue file name
file_catalogue = 'mpcorb.dat.gz'

# output file name
file_output = 'ai202209_s10_00_04.png'

#
# parameters for histogram
#

# width of a bin in au
bin_width = 0.1

# number of bins
n_bins = 100

# bins
bins = numpy.linspace (0.0, bin_width * (n_bins - 1), n_bins)

# making a numpy array for histogram
hist_a = numpy.zeros (n_bins, dtype=int)

# opening catalogue file
with gzip.open (file_catalogue, 'rb') as fh:
    # reading catalogue line-by-line
    for line in fh:
        # number of provisional designation

```

```
try:
    designation = line[0:7].strip ().decode ('utf-8')
except:
    continue
# absolute magnitude
try:
    absmag = float (line[8:13])
except:
    absmag = -999.99
# mean anomaly
try:
    M = float (line[26:35])
except:
    M = -999.99
# argument of perihelion
try:
    peri = float (line[37:46])
except:
    peri = -999.99
# longitude of ascending node
try:
    node = float (line[48:57])
except:
    node = -999.99
# inclination
try:
    incl = float (line[59:68])
except:
    incl = -999.99
# eccentricity
try:
    e = float (line[70:79])
except:
    e = -999.99
# semimajor axis
try:
    a = float (line[92:103])
except:
    a = -999.99
# number of observations
try:
    nobs = int (line[117:122])
except:
    nobs = -999
# residual
try:
    residual = float (line[137:141])
except:
    residual = -999.99
# 4-hexdigit flags
try:
    flag = line[161:165].strip ().decode ('utf-8')
except:
    flag = '9999'
# readable name
try:
    name = line[166:194].strip ().decode ('utf-8')
except:
    name = '__NONE__'
```

```

# last observation date
try:
    lastobs = int (line[194:202])
except:
    lastobs = 99999999

# skip when reading the header
if ( (a < -999.0) and (e < -999.0) and (incl < -999.0) \
      and (peri < -999.0) and (node < -999.0) and (M < -999.0) ):
    continue

# counting number of asteroids in each bin
if (a * 10 < n_bins):
    i = int (a * 10)
    hist_a[i] += 1

# printing histogram
for i in range (n_bins):
    start = i * bin_width
    end   = (i + 1) * bin_width
    print (f"from {start:5.2f} au to {end:5.2f} au : {hist_a[i]:8d}")

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

# labels
ax.set_xlabel ('Semimajor axis [au]')
ax.set_ylabel ('Number of asteroids')

# axes
ax.set_xlim (0.0, 10.0)
ax.set_xticks (numpy.linspace (0.0, 10.0, 11))

# plotting a histogram
ax.bar (bins, height=hist_a, width=bin_width, align='edge')

# saving the plot into a file
fig.savefig (file_output, dpi=225)

```

Execute above script to read the file “MPCORB.DAT”, construct a histogram of semimajor axis, and make a plot using Matplotlib.

```

% chmod a+x ai202209_s10_00_04.py
% ./ai202209_s10_00_04.py
from 0.00 au to 0.10 au :      0
from 0.10 au to 0.20 au :      0
from 0.20 au to 0.30 au :      0
from 0.30 au to 0.40 au :      0
from 0.40 au to 0.50 au :       1
from 0.50 au to 0.60 au :       5
from 0.60 au to 0.70 au :      58
from 0.70 au to 0.80 au :     308
from 0.80 au to 0.90 au :     823
from 0.90 au to 1.00 au :    1222

.....

```

```

from 9.00 au to 9.10 au :      4
from 9.10 au to 9.20 au :      3
from 9.20 au to 9.30 au :      4
from 9.30 au to 9.40 au :      4
from 9.40 au to 9.50 au :      1
from 9.50 au to 9.60 au :      2
from 9.60 au to 9.70 au :      6
from 9.70 au to 9.80 au :      5
from 9.80 au to 9.90 au :      3
from 9.90 au to 10.00 au :     3

```

Display the PNG file. (Fig. 7) Majority of asteroids are located between 2.0 au and 3.5 au from the Sun.

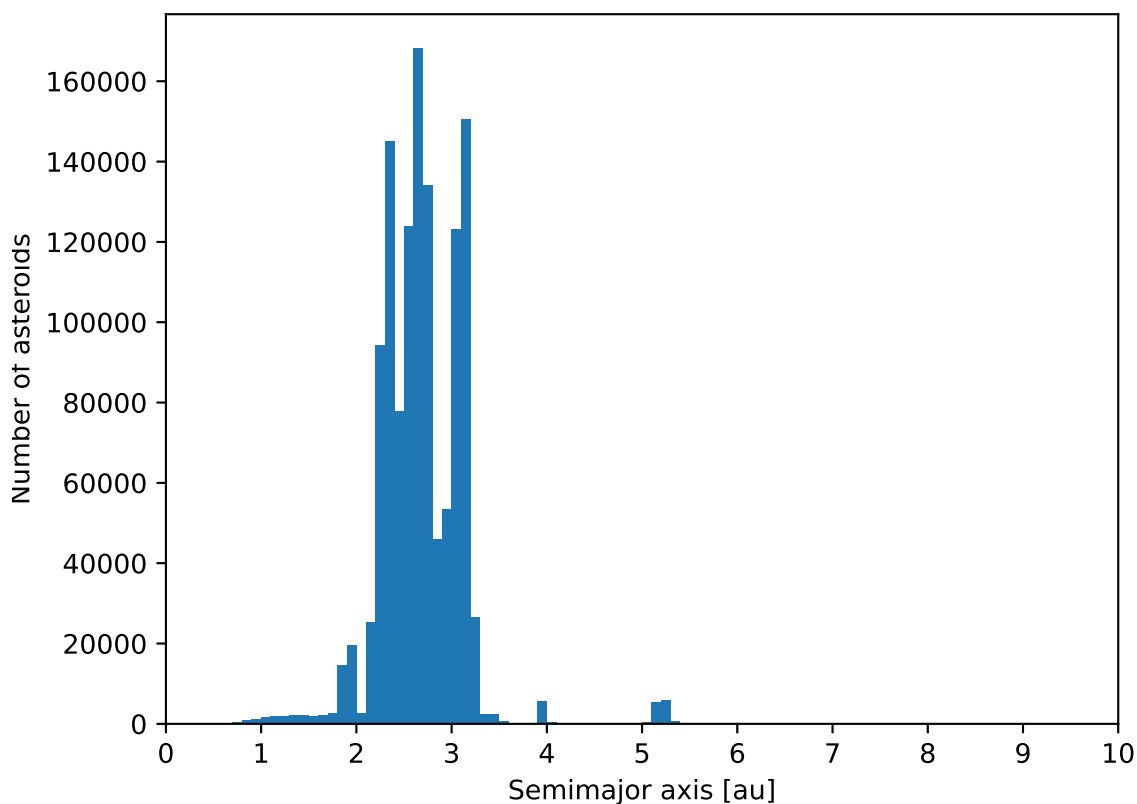


Figure 7: The histogram of semimajor axis of asteroids.

Try following practice.

Practice 10-05

Read the file “MPCORB.DAT”, construct a histogram of eccentricity of asteroids, and make a plot.

2.4 Inclination distribution of asteroids

Construct a histogram of inclination of asteroids and make a plot. Here is a sample script.

Python Code 6: ai202209_s10_00_05.py

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

```
#
# Time-stamp: <2022/11/18 23:00:49 (CST) daisuke>
#

# importing gzip module
import gzip

# importing numpy module
import numpy

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

# catalogue file name
file_catalogue = 'mpcorb.dat.gz'

# output file name
file_output = 'ai202209_s10_00_05.png'

#
# parameters for histogram
#

# width of a bin in au
bin_width = 1.0

# number of bins
n_bins = 180

# bins
bins = numpy.linspace (0.0, bin_width * (n_bins - 1), n_bins)

# making a numpy array for histogram
hist_incl = numpy.zeros (n_bins, dtype=int)

# opening catalogue file
with gzip.open (file_catalogue, 'rb') as fh:
    # reading catalogue line-by-line
    for line in fh:
        # number of provisional designation
        try:
            designation = line[0:7].strip ().decode ('utf-8')
        except:
            continue
        # absolute magnitude
        try:
            absmag = float (line[8:13])
        except:
            absmag = -999.99
        # mean anomaly
        try:
            M = float (line[26:35])
        except:
            M = -999.99
        # argument of perihelion
        try:
            peri = float (line[37:46])
        except:
```

```

        peri = -999.99
# longitude of ascending node
try:
    node = float (line[48:57])
except:
    node = -999.99
# inclination
try:
    incl = float (line[59:68])
except:
    incl = -999.99
# eccentricity
try:
    e = float (line[70:79])
except:
    e = -999.99
# semimajor axis
try:
    a = float (line[92:103])
except:
    a = -999.99
# number of observations
try:
    nobs = int (line[117:122])
except:
    nobs = -999
# residual
try:
    residual = float (line[137:141])
except:
    residual = -999.99
# 4-hexdigit flags
try:
    flag = line[161:165].strip ().decode ('utf-8')
except:
    flag = '9999'
# readable name
try:
    name = line[166:194].strip ().decode ('utf-8')
except:
    name = '__NONE__'
# last observation date
try:
    lastobs = int (line[194:202])
except:
    lastobs = 99999999

# skip when reading the header
if ( (a < -999.0) and (e < -999.0) and (incl < -999.0) \
    and (peri < -999.0) and (node < -999.0) and (M < -999.0) ):
    continue

# counting number of asteroids in each bin
i = int (incl)
hist_incl[i] += 1

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

```

```

ax      = fig.add_subplot (111)

# labels
ax.set_xlabel ('Inclination [deg]')
ax.set_ylabel ('Number of asteroids')

# axes
ax.set_xlim (0.0, 90.0)
ax.set_xticks (numpy.linspace (0.0, 90.0, 10))

# plotting a histogram
ax.bar (bins, height=hist_incl, width=bin_width, align='edge')

# saving the plot into a file
fig.savefig (file_output, dpi=225)

```

Execute above script to read the file “MPCORB.DAT”, construct a histogram of inclination, and make a plot using Matplotlib.

```

% chmod a+x ai202209_s10_00_05.py
% ./ai202209_s10_00_05.py

```

Display the PNG file. (Fig. 8) Many asteroids have small inclination angle, and hence many asteroids are located near the Ecliptic plane.

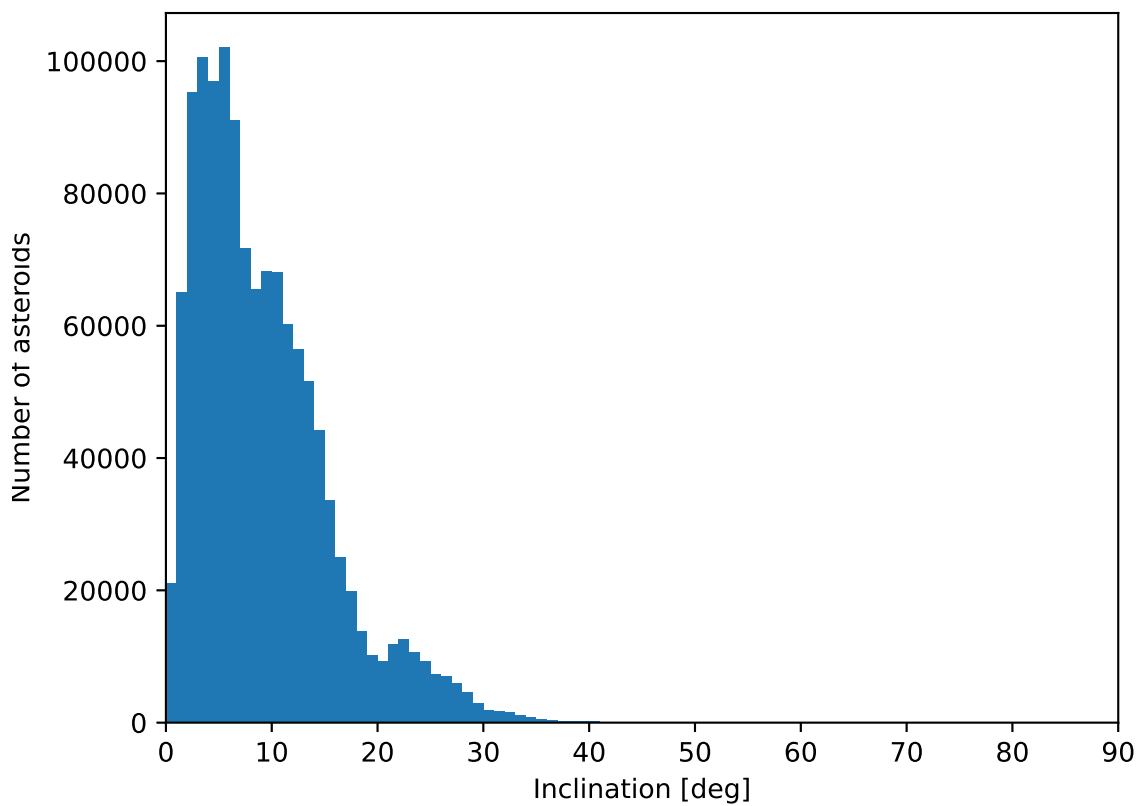


Figure 8: The histogram of argument of perihelion of asteroids.

Try following practice.

Practice 10-06

Read the file “MPCORB.DAT”, construct a histogram of eccentricity of asteroids, and make a plot.

2.5 Absolute magnitude distribution of asteroids

Construct a histogram of absolute magnitude of asteroids and make a plot. Here is a sample script.

Python Code 7: ai202209_s10_00_06.py

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

#
# Time-stamp: <2022/11/18 23:01:15 (CST) daisuke>
#

# importing gzip module
import gzip

# importing numpy module
import numpy

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

# catalogue file name
file_catalogue = 'mpcorb.dat.gz'

# output file name
file_output = 'ai202209_s10_00_06.png'

#
# parameters for histogram
#

# width of a bin in au
bin_width = 0.5

# number of bins
n_bins = 100

# bins
bins = numpy.linspace (-10.0, -10.0 + bin_width * (n_bins - 1), n_bins)

# making a numpy array for histogram
hist_absmag = numpy.zeros (n_bins, dtype=int)

# opening catalogue file
with gzip.open (file_catalogue, 'rb') as fh:
    # reading catalogue line-by-line
    for line in fh:
        # number of provisional designation
        try:
            designation = line[0:7].strip ().decode ('utf-8')
        except:
            continue
        # absolute magnitude
        try:
            absmag = float (line[8:13])
```

```
except:
    absmag = -999.99
# mean anomaly
try:
    M = float (line[26:35])
except:
    M = -999.99
# argument of perihelion
try:
    peri = float (line[37:46])
except:
    peri = -999.99
# longitude of ascending node
try:
    node = float (line[48:57])
except:
    node = -999.99
# inclination
try:
    incl = float (line[59:68])
except:
    incl = -999.99
# eccentricity
try:
    e = float (line[70:79])
except:
    e = -999.99
# semimajor axis
try:
    a = float (line[92:103])
except:
    a = -999.99
# number of observations
try:
    nobs = int (line[117:122])
except:
    nobs = -999
# residual
try:
    residual = float (line[137:141])
except:
    residual = -999.99
# 4-hexdigit flags
try:
    flag = line[161:165].strip ().decode ('utf-8')
except:
    flag = '9999'
# readable name
try:
    name = line[166:194].strip ().decode ('utf-8')
except:
    name = '__NONE__'
# last observation date
try:
    lastobs = int (line[194:202])
except:
    lastobs = 99999999

# skip when reading the header
```

```

        if ( (a < -999.0) and (e < -999.0) and (incl < -999.0) \
            and (peri < -999.0) and (node < -999.0) and (M < -999.0) ):
            continue

        # counting number of asteroids in each bin
        if ( (absmag >= -10.0) and (absmag < 40.0) ):
            i = int ( (absmag + 10.0) * 2.0 )
            hist_absmag[i] += 1

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

# labels
ax.set_xlabel ('Absolute magnitude [mag]')
ax.set_ylabel ('Number of asteroids')

# axes
ax.set_xlim (-10.0, 40.0)
ax.set_xticks (numpy.linspace (-10.0, 40.0, 11))

# plotting a histogram
ax.bar (bins, height=hist_absmag, width=bin_width, align='edge')

# saving the plot into a file
fig.savefig (file_output, dpi=225)

```

Execute above script to read the file “MPCORB.DAT”, construct a histogram of absolute magnitude, and make a plot using Matplotlib.

```

% chmod a+x ai202209_s10_00_06.py
% ./ai202209_s10_00_06.py

```

Display the PNG file. (Fig. 9)
Try following practice.

Practice 10-07

Read the file “MPCORB.DAT”, construct a histogram of longitude of ascending node of asteroids, and make a plot.

Use logarithmic scale for Y-axis, and re-generate the plot.

Python Code 8: ai202209_s10_00_07.py

```

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

#
# Time-stamp: <2022/11/18 23:18:26 (CST) daisuke>
#

# importing gzip module
import gzip

# importing numpy module
import numpy

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

```

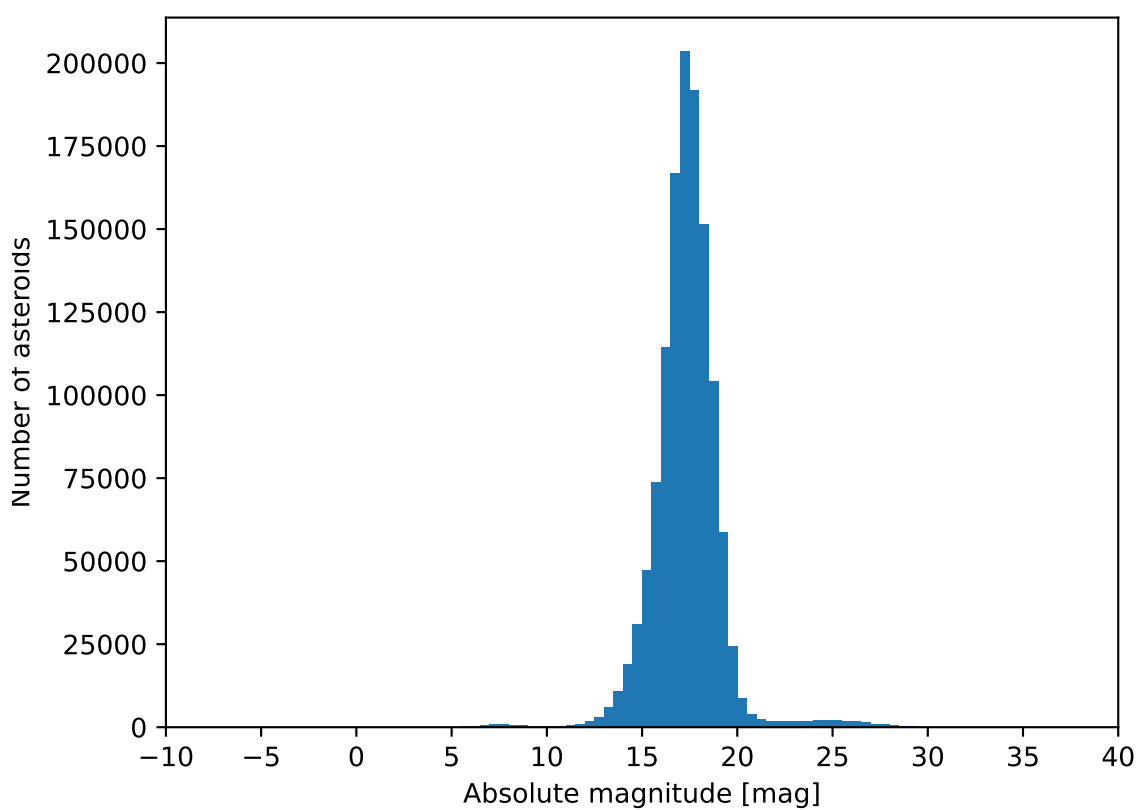


Figure 9: The histogram of absolute magnitude of asteroids.

```
# catalogue file name
file_catalogue = 'mpcorb.dat.gz'

# output file name
file_output = 'ai202209_s10_00_07.png'

#
# parameters for histogram
#

# width of a bin in au
bin_width = 0.5

# number of bins
n_bins = 100

# bins
bins = numpy.linspace (-10.0, -10.0 + bin_width * (n_bins - 1), n_bins)

# making a numpy array for histogram
hist_absmag = numpy.zeros (n_bins, dtype=int)

# opening catalogue file
with gzip.open (file_catalogue, 'rb') as fh:
    # reading catalogue line-by-line
    for line in fh:
        # number of provisional designation
        try:
            designation = line[0:7].strip ().decode ('utf-8')
        except:
            continue
        # absolute magnitude
        try:
            absmag = float (line[8:13])
        except:
            absmag = -999.99
        # mean anomaly
        try:
            M = float (line[26:35])
        except:
            M = -999.99
        # argument of perihelion
        try:
            peri = float (line[37:46])
        except:
            peri = -999.99
        # longitude of ascending node
        try:
            node = float (line[48:57])
        except:
            node = -999.99
        # inclination
        try:
            incl = float (line[59:68])
        except:
            incl = -999.99
        # eccentricity
        try:
```

```

        e = float (line[70:79])
    except:
        e = -999.99
    # semimajor axis
    try:
        a = float (line[92:103])
    except:
        a = -999.99
    # number of observations
    try:
        nobs = int (line[117:122])
    except:
        nobs = -999
    # residual
    try:
        residual = float (line[137:141])
    except:
        residual = -999.99
    # 4-hexdigit flags
    try:
        flag = line[161:165].strip ().decode ('utf-8')
    except:
        flag = '9999'
    # readable name
    try:
        name = line[166:194].strip ().decode ('utf-8')
    except:
        name = '__NONE__'
    # last observation date
    try:
        lastobs = int (line[194:202])
    except:
        lastobs = 99999999

    # skip when reading the header
    if ( (a < -999.0) and (e < -999.0) and (incl < -999.0) \
        and (peri < -999.0) and (node < -999.0) and (M < -999.0) ):
        continue

    # counting number of asteroids in each bin
    if ( (absmag >= -10.0) and (absmag < 40.0) ):
        i = int ( (absmag + 10.0) * 2.0 )
        hist_absmag[i] += 1

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

# labels
ax.set_xlabel ('Absolute magnitude [mag]')
ax.set_ylabel ('Number of asteroids')

# axes
ax.set_xlim (-10.0, 40.0)
ax.set_xticks (numpy.linspace (-10.0, 40.0, 11))
ax.set_yscale ('log')
ax.set_ylim (0.5, 10**6)

```

```
# plotting a histogram
ax.bar (bins, height=hist_absmag, width=bin_width, align='edge')

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to read the file “MPCORB.DAT”, construct a histogram of absolute magnitude, and make a plot using Matplotlib.

```
% chmod a+x ai202209_s10_00_07.py
% ./ai202209_s10_00_07.py
```

Display the PNG file. (Fig. 10)

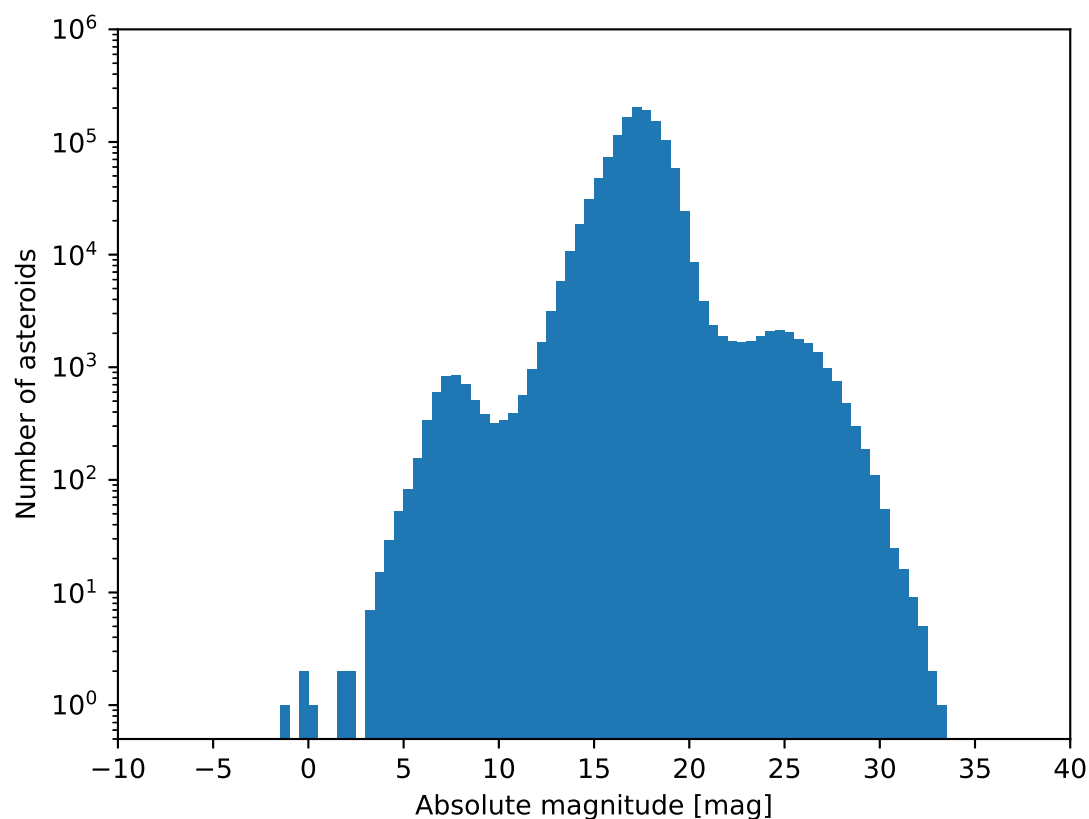


Figure 10: The histogram of absolute magnitude of asteroids.

Try following practice.

Practice 10-08

Read the file “MPCORB.DAT”, construct a histogram of mean anomaly of asteroids, and make a plot.

2.6 Getting asteroid position

Use the package `astroquery` to fetch the position of an asteroid. Here is an example for getting the position of (1) Ceres on 01/Jan/2023.

Python Code 9: ai202209_s10_00_08.py

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

#
# Time-stamp: <2022/11/20 13:03:17 (CST) daisuke>
#

# importing astropy module
import astropy.time

# importing astroquery module
import astroquery.jplhorizons

# date/time
date = astropy.time.Time ('2023-01-01 00:00:00')

# target asteroid name
asteroid = 'Ceres'

# set-up a query for JPL Horizons
query = astroquery.jplhorizons.Horizons (id=asteroid, \
                                         id_type='smallbody', \
                                         epochs=date.jd)

# fetching ephemeris of asteroid
eph = query.ephemerides ()

# printing position of asteroid
print (f"Ephemeris of asteroid '{eph['targetname'][0]}':")
print (f"   RA  = {eph['RA'][0]:8.4f} deg")
print (f"   Dec = {eph['DEC'][0]:+8.4f} deg")
```

Execute above script to get the position of (1) Ceres on 01/Jan/2023.

```
% chmod a+x ai202209_s10_00_08.py
% ./ai202209_s10_00_08.py
Ephemeris of asteroid '1 Ceres (A801 AA)':
  RA  = 187.3734 deg
  Dec = +9.7846 deg
```

Try following practice.

Practice 10-09

Get the position of asteroid Vesta on 15/Jun/2023 using `astroquery`.

Use the package `astroquery` to fetch positions of 10 asteroids. Here is an example for getting positions of (1) Ceres, (2) Pallas, (3) Juno, (4) Vesta, *cdots*, (10) Hygiea on 01/Jan/2023.

Python Code 10: ai202209_s10_00_09.py

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

#
# Time-stamp: <2022/11/20 13:10:29 (CST) daisuke>
#

# importing astropy module
import astropy.time
```

```
# importing astroquery module
import astroquery.jplhorizons

# date/time
date = astropy.time.Time ('2023-01-01 00:00:00')

# number of asteroids to get position
n_asteroids = 10

# processing for each asteroid
for i in range (1, n_asteroids + 1):
    # set-up a query for JPL Horizons
    query = astroquery.jplhorizons.Horizons (id=f"{i}", \
                                              id_type='smallbody', \
                                              epochs=date.jd)

    # fetching ephemeris of asteroid
    eph = query.ephemerides ()

    # printing RA and Dec of asteroid
    print (f"{eph['targetname'][0]:24s}:" \
          + f" (RA, Dec) = ({eph['RA'][0]:8.4f} deg," \
          + f" {eph['DEC'][0]:+8.4f} deg)")
```

Execute above script to get the positions of 10 asteroids on 01/Jan/2023.

```
% chmod a+x ai202209_s10_00_09.py
% ./ai202209_s10_00_09.py
1 Ceres (A801 AA) : (RA, Dec) = (187.3734 deg, +9.7846 deg)
2 Pallas (A802 FA) : (RA, Dec) = (103.8890 deg, -32.0147 deg)
3 Juno (A804 RA) : (RA, Dec) = (357.4604 deg, -8.5643 deg)
4 Vesta (A807 FA) : (RA, Dec) = (347.4824 deg, -12.2712 deg)
5 Astraea (A845 XA) : (RA, Dec) = (349.9594 deg, -8.1209 deg)
6 Hebe (A847 NA) : (RA, Dec) = (132.4608 deg, +9.7006 deg)
7 Iris (A847 PA) : (RA, Dec) = (212.8121 deg, -17.8497 deg)
8 Flora (A847 UA) : (RA, Dec) = (267.6340 deg, -21.4327 deg)
9 Metis (A848 HA) : (RA, Dec) = (325.6073 deg, -18.4135 deg)
10 Hygiea (A849 GA) : (RA, Dec) = (275.1425 deg, -23.8649 deg)
```

Try following practice.

Practice 10-10

Get positions of 100 asteroids from asteroid number 1 to asteroid number 100 on 20/Mar/2023 using `astroquery`.

2.7 Plotting positions of asteroids on the sky

Make a Python script to fetch positions of 100 asteroids using `astroquery`, and make a plot of positions of asteroids on the sky. Here is an example.

Python Code 11: ai202209_s10_00_10.py

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

#
# Time-stamp: <2022/11/20 13:52:49 (CST) daisuke>
#
```

```
# importing numpy module
import numpy

# importing astropy module
import astropy.time

# importing astroquery module
import astroquery.jplhorizons

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

# output file name
file_output = 'ai202209_s10_00_10.png'

# date/time
date = astropy.time.Time ('2023-01-01 00:00:00')

# number of asteroids to get position
n_asteroids = 100

# making empty numpy arrays for storing data
data_ra_hr = numpy.array ([])
data_dec_deg = numpy.array ([])

# processing for each asteroid
for i in range (1, n_asteroids + 1):
    # set-up a query for JPL Horizons
    query = astroquery.jplhorizons.Horizons (id=f"{i}", \
                                              id_type='smallbody', \
                                              epochs=date.jd)

    # fetching ephemeris of asteroid
    eph = query.ephemerides ()

    # printing RA and Dec of asteroid
    print (f"{eph['targetname'][0]:24s}:" \
          + f" (RA, Dec) = ({eph['RA'][0]:8.4f} deg," \
          + f" {eph['DEC'][0]:+8.4f} deg)")

    # RA in hour
    ra_hr = eph['RA'][0] / 15.0

    # Dec in deg
    dec_deg = eph['DEC'][0]

    # appending data to numpy arrays
    data_ra_hr = numpy.append (data_ra_hr, ra_hr)
    data_dec_deg = numpy.append (data_dec_deg, dec_deg)

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

# axes
ax.grid ()
ax.set_xlim (24.0, 0.0)
```

```

ax.set_ylim (-90.0, +90.0)
ax.set_xticks (numpy.linspace (0, 24, 9))
ax.set_yticks (numpy.linspace (-90, 90, 7))
ax.set_xlabel ('Right Ascension [hr]')
ax.set_ylabel ('Declination [deg]')

# title
text_title = f"Distribution of asteroids on {date}"
ax.set_title (text_title)

# plotting data
ax.plot (data_ra_hr, data_dec_deg, \
        linestyle='None', marker='o', color='b', markersize=3, alpha=0.3, \
        label='Asteroids')
ax.legend ()

# saving file
fig.savefig (file_output, dpi=225)

```

Execute above script to make a plot of positions of 100 asteroids on the sky on 01/Jan/2023.

```

% chmod a+x ai202209_s10_00_10.py
% ./ai202209_s10_00_10.py
1 Ceres (A801 AA)      : (RA, Dec) = (187.3734 deg, +9.7846 deg)
2 Pallas (A802 FA)    : (RA, Dec) = (103.8890 deg, -32.0147 deg)
3 Juno (A804 RA)      : (RA, Dec) = (357.4604 deg, -8.5643 deg)
4 Vesta (A807 FA)     : (RA, Dec) = (347.4824 deg, -12.2712 deg)
5 Astraea (A845 XA)   : (RA, Dec) = (349.9594 deg, -8.1209 deg)
6 Hebe (A847 NA)      : (RA, Dec) = (132.4608 deg, +9.7006 deg)
7 Iris (A847 PA)      : (RA, Dec) = (212.8121 deg, -17.8497 deg)
8 Flora (A847 UA)     : (RA, Dec) = (267.6340 deg, -21.4327 deg)
9 Metis (A848 HA)     : (RA, Dec) = (325.6073 deg, -18.4135 deg)
10 Hygiea (A849 GA)   : (RA, Dec) = (275.1425 deg, -23.8649 deg)
...

91 Aegina (A866 VA)   : (RA, Dec) = (231.1236 deg, -19.5023 deg)
92 Undina (A867 NA)   : (RA, Dec) = ( 27.3632 deg, +0.8835 deg)
93 Minerva (A867 QA)  : (RA, Dec) = (326.7358 deg, -16.4782 deg)
94 Aurora (A867 RA)   : (RA, Dec) = (252.0405 deg, -28.2241 deg)
95 Arethusa (A867 WA) : (RA, Dec) = (327.9617 deg, -1.6566 deg)
96 Aegle (A868 DA)    : (RA, Dec) = (179.8819 deg, -6.5820 deg)
97 Klotho (A868 DB)   : (RA, Dec) = (294.7433 deg, -15.4742 deg)
98 Ianthe (A868 HA)   : (RA, Dec) = (247.3587 deg, -32.8008 deg)
99 Dike (A868 KA)     : (RA, Dec) = ( 44.2550 deg, +23.7050 deg)
100 Hekate (A868 NA)  : (RA, Dec) = ( 93.4510 deg, +18.6434 deg)

```

Display the PNG file. (Fig. 11)

```
% feh -dF ai202209_s10_00_10.png
```

Try following practice.

Practice 10-11

Get positions of 300 asteroids from asteroid number 1 to asteroid number 300 on 10/May/2023 using `astroquery`, and make a plot of asteroid positions on the sky on 10/May/2023.

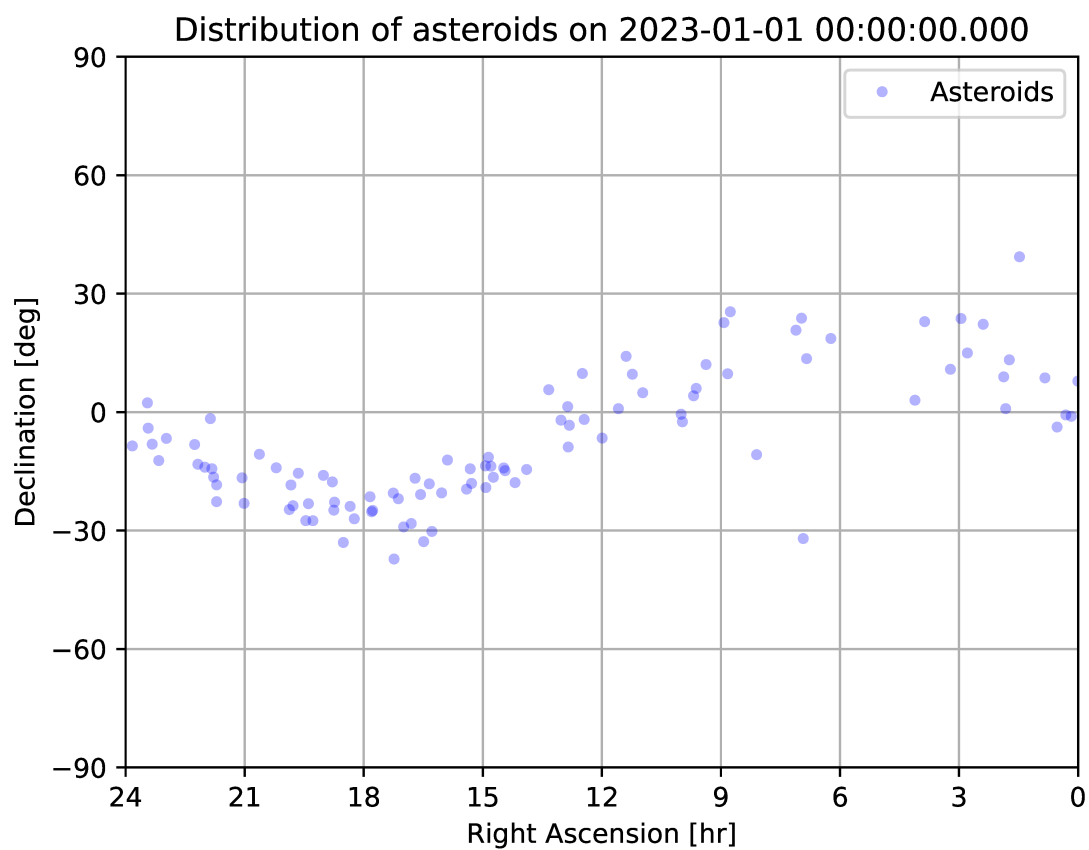


Figure 11: Positions of 100 asteroids on the sky on 01/Jan/2023.

2.8 Dawing the position of the Ecliptic plane

Make a Python script to fetch positions of 300 asteroids using `astroquery`, and make a plot of positions of asteroids on the sky. Superimpose the location of the Ecliptic plane. Here is an example.

Python Code 12: ai202209_s10_00_11.py

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

#
# Time-stamp: <2022/11/20 14:00:14 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.coordinates
import astropy.time
import astropy.units

# importing astroquery module
import astroquery.jplhorizons

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

# output file name
file_output = 'ai202209_s10_00_11.png'

# units
u_deg = astropy.units.degree

# date/time
date = astropy.time.Time ('2023-01-01 00:00:00')

# number of asteroids to get position
n_asteroids = 300

# making empty numpy arrays for storing data
data_ra_hr = numpy.array ([])
data_dec_deg = numpy.array ([])

# processing for each asteroid
for i in range (1, n_asteroids + 1):
    # set-up a query for JPL Horizons
    query = astroquery.jplhorizons.Horizons (id=f"{i}", \
                                              id_type='smallbody', \
                                              epochs=date.jd)

    # fetching ephemeris of asteroid
    eph = query.ephemerides ()

    # printing RA and Dec of asteroid
    print (f"{eph['targetname'][0]:24s}:" \
          + f" (RA, Dec) = ({eph['RA'][0]:8.4f} deg," \
          + f" {eph['DEC'][0]:+8.4f} deg)")

    # RA in hour
    ra_hr = eph['RA'][0] / 15.0
```

```

# Dec in deg
dec_deg = eph['DEC'][0]

# appending data to numpy arrays
data_ra_hr = numpy.append (data_ra_hr, ra_hr)
data_dec_deg = numpy.append (data_dec_deg, dec_deg)

# ecliptic plane
ecl_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
ecl_lat = numpy.zeros (1000) * u_deg
ecl_coord = astropy.coordinates.GeocentricMeanEcliptic (lon=ecl_lon, \
                                                         lat=ecl_lat, \
                                                         obstime=date)
ecl_ra = ecl_coord.transform_to (astropy.coordinates.ICRS).ra.deg / 15.0
ecl_dec = ecl_coord.transform_to (astropy.coordinates.ICRS).dec.deg

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

# axes
ax.grid ()
ax.set_xlim (24.0, 0.0)
ax.set_ylim (-90.0, +90.0)
ax.set_xticks (numpy.linspace (0, 24, 9))
ax.set_yticks (numpy.linspace (-90, 90, 7))
ax.set_xlabel ('Right Ascension [hr]')
ax.set_ylabel ('Declination [deg]')

# title
text_title = f"Distribution of asteroids on {date}"
ax.set_title (text_title)

# plotting data
ax.plot (ecl_ra, ecl_dec, \
        linestyle='None', marker='o', color='y', markersize=5, alpha=0.5, \
        label='Ecliptic plane')
ax.plot (data_ra_hr, data_dec_deg, \
        linestyle='None', marker='o', color='b', markersize=3, alpha=0.3, \
        label='Asteroids')
ax.legend ()

# saving file
fig.savefig (file_output, dpi=225)

```

Execute above script to make a plot of positions of 300 asteroids on the sky on 01/Jan/2023.

```

% chmod a+x ai202209_s10_00_11.py
% ./ai202209_s10_00_11.py
1 Ceres (A801 AA)      : (RA, Dec) = (187.3734 deg, +9.7846 deg)
2 Pallas (A802 FA)    : (RA, Dec) = (103.8890 deg, -32.0147 deg)
3 Juno (A804 RA)      : (RA, Dec) = (357.4604 deg, -8.5643 deg)
4 Vesta (A807 FA)     : (RA, Dec) = (347.4824 deg, -12.2712 deg)
5 Astraea (A845 XA)   : (RA, Dec) = (349.9594 deg, -8.1209 deg)
6 Hebe (A847 NA)      : (RA, Dec) = (132.4608 deg, +9.7006 deg)
7 Iris (A847 PA)      : (RA, Dec) = (212.8121 deg, -17.8497 deg)
8 Flora (A847 UA)     : (RA, Dec) = (267.6340 deg, -21.4327 deg)

```

```

9 Metis (A848 HA)      : (RA, Dec) = (325.6073 deg, -18.4135 deg)
10 Hygiea (A849 GA)   : (RA, Dec) = (275.1425 deg, -23.8649 deg)
...
291 Alice (A890 HA)    : (RA, Dec) = (248.0807 deg, -20.5668 deg)
292 Ludovica (A890 HB) : (RA, Dec) = (225.2810 deg, -13.7545 deg)
293 Brasilia (A890 KA) : (RA, Dec) = (354.9084 deg, -15.2366 deg)
294 Felicia (A890 NA)  : (RA, Dec) = (217.0232 deg, -9.1085 deg)
295 Theresia (A890 QA) : (RA, Dec) = (133.4150 deg, +15.7259 deg)
296 Phaetusa (A890 QB) : (RA, Dec) = (254.3814 deg, -21.5901 deg)
297 Caecilia (A890 RA) : (RA, Dec) = (151.1199 deg, +14.5604 deg)
298 Baptistina (A890 RB): (RA, Dec) = (307.5037 deg, -22.8989 deg)
299 Thora (A890 TA)    : (RA, Dec) = (306.8290 deg, -17.9707 deg)
300 Geraldina (A890 TB) : (RA, Dec) = (347.5120 deg, -5.8168 deg)

```

Display the PNG file. (Fig. 12) Majority of asteroids are found near the Ecliptic plane.

```
% feh -dF ai202209_s10_00_11.png
```

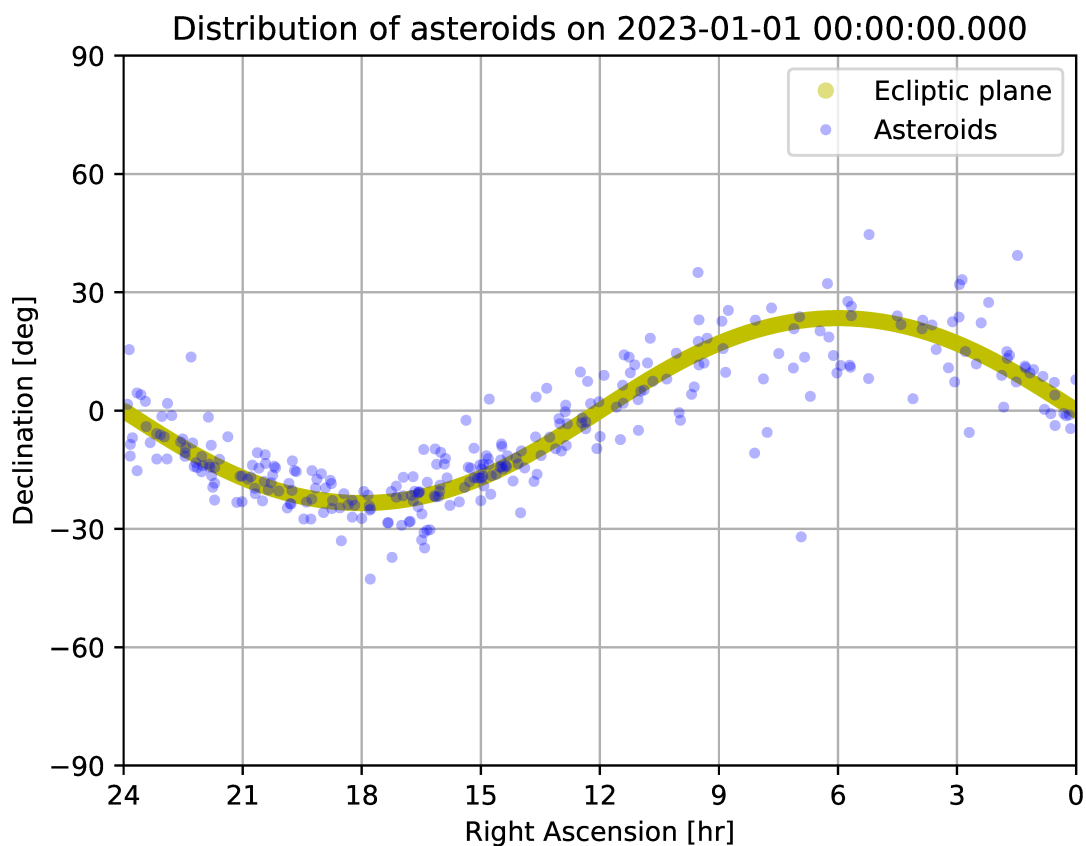


Figure 12: Positions of 300 asteroids on the sky on 01/Jan/2023. The location of the Ecliptic plane is superimposed on the plot.

Try following practice.

Practice 10-12

Get positions of 500 asteroids from asteroid number 1 to asteroid number 500 on 20/Jul/2023 using `astroquery`, and make a plot of asteroid positions on the sky on 10/May/2023. Superimpose the location of the Ecliptic plane on the plot.

2.9 Dawing the position of the Galactic plane

Make a Python script to fetch positions of 500 asteroids using `astroquery`, and make a plot of positions of asteroids on the sky. Superimpose the location of the Ecliptic plane and the Galactic plane on the plot. Here is an example.

Python Code 13: ai202209_s10_00_12.py

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

#
# Time-stamp: <2022/11/20 20:17:18 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.coordinates
import astropy.time
import astropy.units

# importing astroquery module
import astroquery.jplhorizons

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

# output file name
file_output = 'ai202209_s10_00_12.png'

# units
u_deg = astropy.units.degree

# date/time
date = astropy.time.Time ('2023-01-01 00:00:00')

# number of asteroids to get position
n_asteroids = 500

# making empty numpy arrays for storing data
data_ra_hr = numpy.array ([])
data_dec_deg = numpy.array ([])

# processing for each asteroid
for i in range (1, n_asteroids + 1):
    # set-up a query for JPL Horizons
    query = astroquery.jplhorizons.Horizons (id=f"{i}", \
                                              id_type='smallbody', \
                                              epochs=date.jd)

    # fetching ephemeris of asteroid
    eph = query.ephemerides ()
```

```

# printing RA and Dec of asteroid
print (f"{eph['targetname'][0]:24s}:" \
      + f" (RA, Dec) = ({eph['RA'][0]:8.4f} deg," \
      + f" {eph['DEC'][0]:+8.4f} deg)")

# RA in hour
ra_hr = eph['RA'][0] / 15.0

# Dec in deg
dec_deg = eph['DEC'][0]

# appending data to numpy arrays
data_ra_hr = numpy.append (data_ra_hr, ra_hr)
data_dec_deg = numpy.append (data_dec_deg, dec_deg)

# ecliptic plane
ecl_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
ecl_lat = numpy.zeros (1000) * u_deg
ecl_coord = astropy.coordinates.GeocentricMeanEcliptic (lon=ecl_lon, \
                                                         lat=ecl_lat, \
                                                         obstime=date)
ecl_ra = ecl_coord.transform_to (astropy.coordinates.ICRS).ra.deg / 15.0
ecl_dec = ecl_coord.transform_to (astropy.coordinates.ICRS).dec.deg

# galactic plane
gal_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
gal_lat = numpy.zeros (1000) * u_deg
gal_coord = astropy.coordinates.Galactic (l=gal_lon, \
                                          b=gal_lat)
gal_ra = gal_coord.transform_to (astropy.coordinates.ICRS).ra.deg / 15.0
gal_dec = gal_coord.transform_to (astropy.coordinates.ICRS).dec.deg

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

# axes
ax.grid ()
ax.set_xlim (24.0, 0.0)
ax.set_ylim (-90.0, +90.0)
ax.set_xticks (numpy.linspace (0, 24, 9))
ax.set_yticks (numpy.linspace (-90, 90, 7))
ax.set_xlabel ('Right Ascension [hr]')
ax.set_ylabel ('Declination [deg]')

# title
text_title = f"Distribution of asteroids on {date}"
ax.set_title (text_title)

# plotting data
ax.plot (ecl_ra, ecl_dec, \
        linestyle='None', marker='o', color='y', markersize=5, alpha=0.5, \
        label='Ecliptic plane')
ax.plot (gal_ra, gal_dec, \
        linestyle='None', marker='o', color='silver', markersize=5, \
        alpha=0.5, label='Galactic plane')
ax.plot (data_ra_hr, data_dec_deg, \
        linestyle='None', marker='o', color='b', markersize=3, alpha=0.3, \

```

```

        label='Asteroids')
ax.legend ()

# saving file
fig.savefig (file_output , dpi=225)

```

Execute above script to make a plot of positions of 500 asteroids on the sky on 01/Jan/2023.

```

% chmod a+x ai202209_s10_00_12.py
% ./ai202209_s10_00_12.py
1 Ceres (A801 AA)      : (RA, Dec) = (187.3734 deg, +9.7846 deg)
2 Pallas (A802 FA)    : (RA, Dec) = (103.8890 deg, -32.0147 deg)
3 Juno (A804 RA)      : (RA, Dec) = (357.4604 deg, -8.5643 deg)
4 Vesta (A807 FA)     : (RA, Dec) = (347.4824 deg, -12.2712 deg)
5 Astraea (A845 XA)   : (RA, Dec) = (349.9594 deg, -8.1209 deg)
6 Hebe (A847 NA)      : (RA, Dec) = (132.4608 deg, +9.7006 deg)
7 Iris (A847 PA)      : (RA, Dec) = (212.8121 deg, -17.8497 deg)
8 Flora (A847 UA)     : (RA, Dec) = (267.6340 deg, -21.4327 deg)
9 Metis (A848 HA)     : (RA, Dec) = (325.6073 deg, -18.4135 deg)
10 Hygiea (A849 GA)   : (RA, Dec) = (275.1425 deg, -23.8649 deg)

...

491 Carina (A902 RG)   : (RA, Dec) = ( 8.8952 deg, -7.0034 deg)
492 Gismonda (A902 RF) : (RA, Dec) = (295.0984 deg, -22.5857 deg)
493 Griseldis (A902 RH) : (RA, Dec) = (282.1105 deg, -34.4828 deg)
494 Virtus (A902 TF)   : (RA, Dec) = (136.9956 deg, +26.7326 deg)
495 Eulalia (A902 UL)  : (RA, Dec) = (262.6194 deg, -21.6610 deg)
496 Gryphia (A902 UM)  : (RA, Dec) = (334.5628 deg, -8.6585 deg)
497 Iva (A902 VB)      : (RA, Dec) = (348.3181 deg, -4.3032 deg)
498 Tokio (A902 XA)    : (RA, Dec) = (316.6774 deg, -22.1779 deg)
499 Venusia (A902 YE)  : (RA, Dec) = (186.2484 deg, -5.0536 deg)
500 Selinur (A903 BJ)  : (RA, Dec) = (233.0830 deg, -26.6636 deg)

```

Display the PNG file. (Fig. 13)

```
% feh -dF ai202209_s10_00_12.png
```

Try following practice.

Practice 10-13

Get positions of 800 asteroids from asteroid number 1 to asteroid number 500 on 01/Sep/2023 using `astroquery`, and make a plot of asteroid positions on the sky on 01/Sep/2023. Superimpose the location of the Ecliptic plane and the Galactic plane on the plot.

2.10 Using Mollweide projection

Make a Python script to fetch positions of 1000 asteroids using `astroquery`, and make a plot of positions of asteroids on the sky using Mollweide projection. Superimpose the location of the Ecliptic plane and the Galactic plane on the plot. Here is an example.

Python Code 14: ai202209_s10_00_13.py

```

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

#
# Time-stamp: <2022/11/20 20:17:30 (CST) daisuke>

```

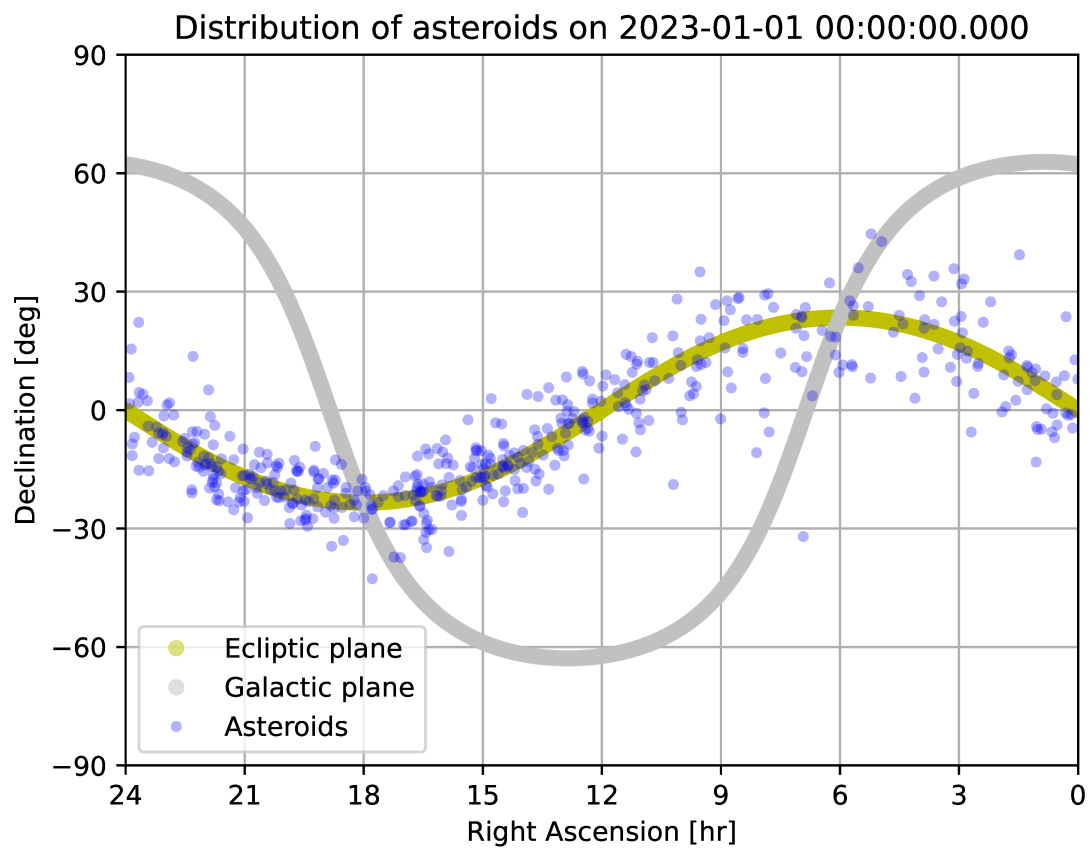


Figure 13: Positions of 300 asteroids on the sky on 01/Jan/2023. The location of the Ecliptic plane and the Galactic plane are superimposed on the plot.

```
#

# importing numpy module
import numpy

# importing astropy module
import astropy.coordinates
import astropy.time
import astropy.units

# importing astroquery module
import astroquery.jplhorizons

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

# output file name
file_output = 'ai202209_s10_00_13.png'

# units
u_deg = astropy.units.degree

# date/time
date = astropy.time.Time ('2023-01-01 00:00:00')

# number of asteroids to get position
n_asteroids = 1000

# making empty numpy arrays for storing data
data_ra_deg = numpy.array ([])
data_dec_deg = numpy.array ([])

# processing for each asteroid
for i in range (1, n_asteroids + 1):
    # set-up a query for JPL Horizons
    query = astroquery.jplhorizons.Horizons (id=f"{i}", \
                                              id_type='smallbody', \
                                              epochs=date.jd)

    # fetching ephemeris of asteroid
    eph = query.ephemerides ()

    # printing RA and Dec of asteroid
    print (f"{eph['targetname'][0]:24s}:" \
          + f" (RA, Dec) = ({eph['RA'][0]:8.4f} deg," \
          + f" {eph['DEC'][0]:+8.4f} deg)")

    # RA in deg
    ra_deg = eph['RA'][0]
    if (ra_deg > 180.0):
        ra_deg -= 360.0

    # Dec in deg
    dec_deg = eph['DEC'][0]

    # appending data to numpy arrays
    data_ra_deg = numpy.append (data_ra_deg, ra_deg)
    data_dec_deg = numpy.append (data_dec_deg, dec_deg)
```

```

# ecliptic plane
ecl_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
ecl_lat = numpy.zeros (1000) * u_deg
ecl_coord = astropy.coordinates.GeocentricMeanEcliptic (lon=ecl_lon, \
                                                         lat=ecl_lat, \
                                                         obstime=date)

ecl_ra  = ecl_coord.transform_to (astropy.coordinates.ICRS) \
          .ra.wrap_at (180.0 * u_deg).radian
ecl_dec = ecl_coord.transform_to (astropy.coordinates.ICRS).dec.radian

# galactic plane
gal_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
gal_lat = numpy.zeros (1000) * u_deg
gal_coord = astropy.coordinates.Galactic (l=gal_lon, \
                                           b=gal_lat)

gal_ra  = gal_coord.transform_to (astropy.coordinates.ICRS) \
          .ra.wrap_at (180.0 * u_deg).radian
gal_dec = gal_coord.transform_to (astropy.coordinates.ICRS).dec.radian

# making objects "fig" and "ax"
fig      = matplotlib.figure.Figure ()
canvas   = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax       = fig.add_subplot (111, projection='mollweide')

# axes
ax.grid ()
ax.set_xlabel ('Right Ascension [deg]')
ax.set_ylabel ('Declination [deg]')

# title
text_title = f"Distribution of asteroids"
ax.set_title (text_title, loc='right')

# plotting data
ax.plot (ecl_ra, ecl_dec, \
         linestyle='None', marker='o', color='y', markersize=5, alpha=0.5, \
         label='Ecliptic plane')
ax.plot (gal_ra, gal_dec, \
         linestyle='None', marker='o', color='silver', markersize=5, \
         alpha=0.5, label='Galactic plane')
ax.plot (numpy.deg2rad (data_ra_deg), numpy.deg2rad (data_dec_deg), \
         linestyle='None', marker='o', color='b', markersize=3, alpha=0.3, \
         label='Asteroids')
ax.legend (bbox_to_anchor=(0.9, -0.1))

# saving file
fig.savefig (file_output, dpi=225)

```

Execute above script to make a plot of positions of 1000 asteroids on the sky on 01/Jan/2023 using Mollweide projection.

```

% chmod a+x ai202209_s10_00_13.py
% ./ai202209_s10_00_13.py
1 Ceres (A801 AA)      : (RA, Dec) = (187.3734 deg, +9.7846 deg)
2 Pallas (A802 FA)     : (RA, Dec) = (103.8890 deg, -32.0147 deg)
3 Juno (A804 RA)       : (RA, Dec) = (357.4604 deg, -8.5643 deg)
4 Vesta (A807 FA)      : (RA, Dec) = (347.4824 deg, -12.2712 deg)
5 Astraea (A845 XA)    : (RA, Dec) = (349.9594 deg, -8.1209 deg)

```

```

6 Hebe (A847 NA)      : (RA, Dec) = (132.4608 deg, +9.7006 deg)
7 Iris (A847 PA)      : (RA, Dec) = (212.8121 deg, -17.8497 deg)
8 Flora (A847 UA)     : (RA, Dec) = (267.6340 deg, -21.4327 deg)
9 Metis (A848 HA)     : (RA, Dec) = (325.6073 deg, -18.4135 deg)
10 Hygiea (A849 GA)   : (RA, Dec) = (275.1425 deg, -23.8649 deg)
...

991 McDondalda (A922 UC) : (RA, Dec) = ( 0.4742 deg, -1.3428 deg)
992 Swasey (A922 VD)    : (RA, Dec) = ( 26.4188 deg, +8.0363 deg)
993 Moultona (A923 AB)  : (RA, Dec) = (324.6609 deg, -13.3489 deg)
994 Otthild (A923 FB)   : (RA, Dec) = (142.2605 deg, +35.8386 deg)
995 Sternberga (A923 LC): (RA, Dec) = (138.9423 deg, -2.8244 deg)
996 Hilaritas (A923 FC) : (RA, Dec) = (296.6866 deg, -21.5698 deg)
997 Priska (A923 NC)    : (RA, Dec) = (218.6922 deg, -21.6610 deg)
998 Bodea (A923 PC)     : (RA, Dec) = (315.4124 deg, -11.3337 deg)
999 Zachia (A923 PE)    : (RA, Dec) = (177.5297 deg, -8.2167 deg)
1000 Piazzia (A923 PF)  : (RA, Dec) = ( 90.9386 deg, +44.9454 deg)

```

Display the PNG file. (Fig. 14)

```
% feh -dF ai202209_s10_00_13.png
```

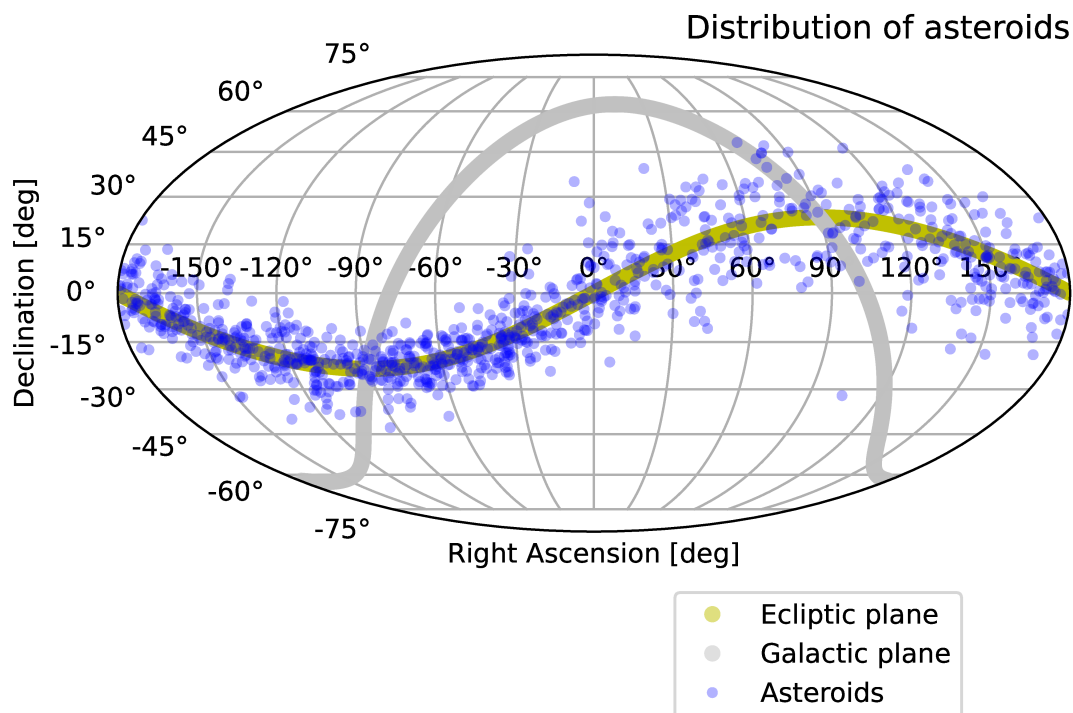


Figure 14: Positions of 1000 asteroids on the sky on 01/Jan/2023. The location of the Ecliptic plane and the Galactic plane are superimposed on the plot.

Try following practice.

Practice 10-14

Get positions of 3000 asteroids from asteroid number 1 to asteroid number 3000 on 15/Oct/2023 using **astroquery**, and make a plot of asteroid positions on the sky on 15/Oct/2023 using Mollweide projection. Superimpose the location of the Ecliptic plane and the Galactic plane on the plot.

2.11 Using Aitoff projection

Make a Python script to fetch positions of 3000 asteroids using **astroquery**, and make a plot of positions of asteroids on the sky using Aitoff projection. Superimpose the location of the Ecliptic plane and the Galactic plane on the plot. Here is an example.

Python Code 15: ai202209_s10_00_14.py

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

#
# Time-stamp: <2022/11/20 20:17:52 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.coordinates
import astropy.time
import astropy.units

# importing astroquery module
import astroquery.jplhorizons

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

# output file name
file_output = 'ai202209_s10_00_14.png'

# units
u_deg = astropy.units.degree

# date/time
date = astropy.time.Time ('2023-01-01 00:00:00')

# number of asteroids to get position
n_asteroids = 3000

# making empty numpy arrays for storing data
data_ra_deg = numpy.array ([])
data_dec_deg = numpy.array ([])

# processing for each asteroid
for i in range (1, n_asteroids + 1):
    # set-up a query for JPL Horizons
    query = astroquery.jplhorizons.Horizons (id=f"{i}", \
                                              id_type='smallbody', \
                                              epochs=date.jd)
```

```

# fetching ephemeris of asteroid
eph = query.ephemerides ()

# printing RA and Dec of asteroid
print (f"{eph['targetname'][0]:24s}:" \
      + f" (RA, Dec) = ({eph['RA'][0]:8.4f} deg," \
      + f" {eph['DEC'][0]:+8.4f} deg)")

# RA in deg
ra_deg = eph['RA'][0]
if (ra_deg > 180.0):
    ra_deg -= 360.0

# Dec in deg
dec_deg = eph['DEC'][0]

# appending data to numpy arrays
data_ra_deg = numpy.append (data_ra_deg, ra_deg)
data_dec_deg = numpy.append (data_dec_deg, dec_deg)

# ecliptic plane
ecl_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
ecl_lat = numpy.zeros (1000) * u_deg
ecl_coord = astropy.coordinates.GeocentricMeanEcliptic (lon=ecl_lon, \
                                                         lat=ecl_lat, \
                                                         obstime=date)

ecl_ra = ecl_coord.transform_to (astropy.coordinates.ICRS) \
        .ra.wrap_at (180.0 * u_deg).radian
ecl_dec = ecl_coord.transform_to (astropy.coordinates.ICRS).dec.radian

# galactic plane
gal_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
gal_lat = numpy.zeros (1000) * u_deg
gal_coord = astropy.coordinates.Galactic (l=gal_lon, \
                                           b=gal_lat)

gal_ra = gal_coord.transform_to (astropy.coordinates.ICRS) \
        .ra.wrap_at (180.0 * u_deg).radian
gal_dec = gal_coord.transform_to (astropy.coordinates.ICRS).dec.radian

# making objects "fig" and "ax"
fig = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax = fig.add_subplot (111, projection='aitoff')

# axes
ax.grid ()
ax.set_xlabel ('Right Ascension [deg]')
ax.set_ylabel ('Declination [deg]')

# title
text_title = f"Distribution of asteroids"
ax.set_title (text_title, loc='right')

# plotting data
ax.plot (ecl_ra, ecl_dec, \
        linestyle='None', marker='o', color='y', markersize=5, alpha=0.5, \
        label='Ecliptic plane')
ax.plot (gal_ra, gal_dec, \
        linestyle='None', marker='o', color='silver', markersize=5, \

```

```

        alpha=0.5, label='Galactic plane')
ax.plot (numpy.deg2rad (data_ra_deg), numpy.deg2rad (data_dec_deg), \
        linestyle='None', marker='o', color='b', markersize=3, alpha=0.3, \
        label='Asteroids')
ax.legend (bbox_to_anchor=(0.9, -0.1))

# saving file
fig.savefig (file_output, dpi=225)

```

Execute above script to make a plot of positions of 3000 asteroids on the sky on 01/Jan/2023 using Aitoff projection.

```

% chmod a+x ai202209_s10_00_14.py
% ./ai202209_s10_00_14.py
1 Ceres (A801 AA)      : (RA, Dec) = (187.3734 deg, +9.7846 deg)
2 Pallas (A802 FA)    : (RA, Dec) = (103.8890 deg, -32.0147 deg)
3 Juno (A804 RA)      : (RA, Dec) = (357.4604 deg, -8.5643 deg)
4 Vesta (A807 FA)     : (RA, Dec) = (347.4824 deg, -12.2712 deg)
5 Astraea (A845 XA)   : (RA, Dec) = (349.9594 deg, -8.1209 deg)
6 Hebe (A847 NA)      : (RA, Dec) = (132.4608 deg, +9.7006 deg)
7 Iris (A847 PA)      : (RA, Dec) = (212.8121 deg, -17.8497 deg)
8 Flora (A847 UA)     : (RA, Dec) = (267.6340 deg, -21.4327 deg)
9 Metis (A848 HA)     : (RA, Dec) = (325.6073 deg, -18.4135 deg)
10 Hygiea (A849 GA)   : (RA, Dec) = (275.1425 deg, -23.8649 deg)
...
2991 Bilbo (1982 HV)   : (RA, Dec) = ( 17.9072 deg, +2.2854 deg)
2992 Vondel (2540 P-L) : (RA, Dec) = (275.9027 deg, -17.8996 deg)
2993 Wendy (1970 PA)   : (RA, Dec) = (207.1615 deg, -22.0275 deg)
2994 Flynn (1975 PA)   : (RA, Dec) = (183.6456 deg, -0.8351 deg)
2995 Taratuta (1978 QK) : (RA, Dec) = (215.2461 deg, -8.1545 deg)
2996 Bowman (1954 RJ)  : (RA, Dec) = (258.2920 deg, -25.7335 deg)
2997 Cabrera (1974 MJ) : (RA, Dec) = (243.2244 deg, -26.0287 deg)
2998 Berendeya (1975 TR3) : (RA, Dec) = (213.1368 deg, -11.4399 deg)
2999 Dante (1981 CY)   : (RA, Dec) = (248.6859 deg, -20.5187 deg)
3000 Leonardo (1981 EG19) : (RA, Dec) = (354.7845 deg, -2.3956 deg)

```

Display the PNG file. (Fig. 15)

```
% feh -dF ai202209_s10_00_14.png
```

Try following practice.

Practice 10-15

Get positions of 5000 asteroids from asteroid number 1 to asteroid number 5000 on 01/Dec/2023 using **astroquery**, and make a plot of asteroid positions on the sky on 01/Dec/2023 using Aitoff projection. Superimpose the location of the Ecliptic plane and the Galactic plane on the plot.

2.12 Using Hammer projection

Make a Python script to fetch positions of 3000 asteroids using **astroquery**, and make a plot of positions of asteroids on the sky using Hammer projection. Superimpose the location of the Ecliptic plane and the Galactic plane on the plot. Here is an example.

Python Code 16: ai202209_s10_00_15.py

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

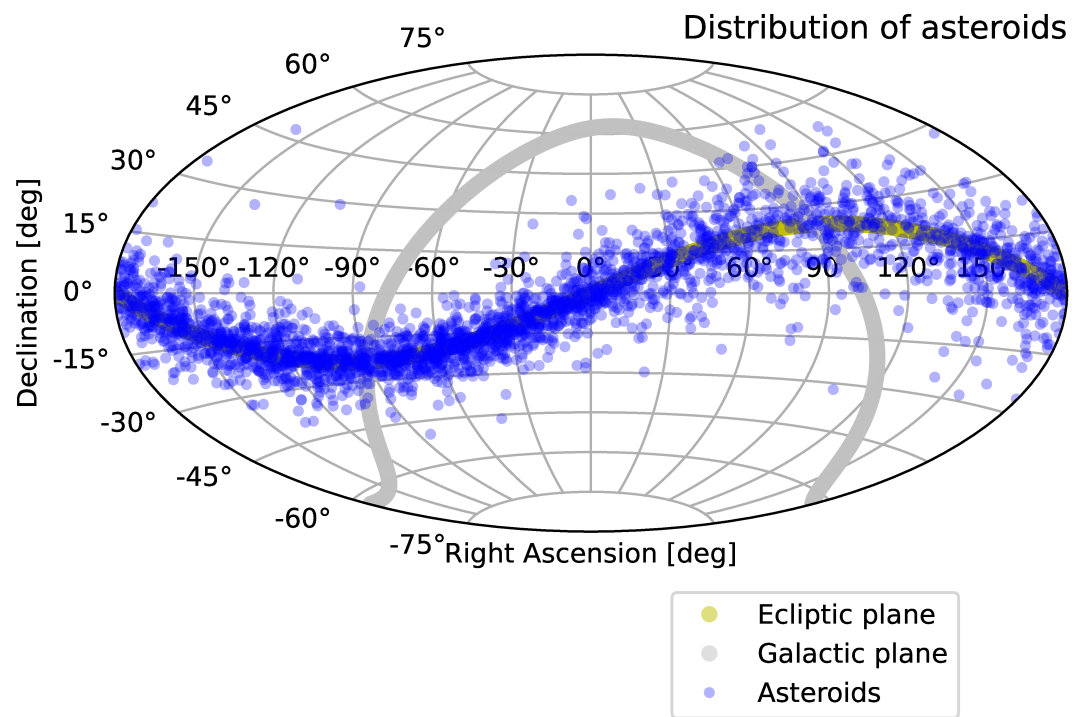


Figure 15: Positions of 3000 asteroids on the sky on 01/Jan/2023 using Aitoff projection. The location of the Ecliptic plane and the Galactic plane are superimposed on the plot.

```
#
# Time-stamp: <2022/11/20 20:18:06 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.coordinates
import astropy.time
import astropy.units

# importing astroquery module
import astroquery.jplhorizons

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

# output file name
file_output = 'ai202209_s10_00_15.png'

# units
u_deg = astropy.units.degree

# date/time
date = astropy.time.Time ('2023-01-01 00:00:00')

# number of asteroids to get position
n_asteroids = 3000

# making empty numpy arrays for storing data
data_ra_deg = numpy.array ([])
data_dec_deg = numpy.array ([])

# processing for each asteroid
for i in range (1, n_asteroids + 1):
    # set-up a query for JPL Horizons
    query = astroquery.jplhorizons.Horizons (id=f"{i}", \
                                              id_type='smallbody', \
                                              epochs=date.jd)

    # fetching ephemeris of asteroid
    eph = query.ephemerides ()

    # printing RA and Dec of asteroid
    print (f"{eph['targetname'][0]:24s}:" \
          + f" (RA, Dec) = ({eph['RA'][0]:8.4f} deg," \
          + f" {eph['DEC'][0]:+8.4f} deg)")

    # RA in deg
    ra_deg = eph['RA'][0]
    if (ra_deg > 180.0):
        ra_deg -= 360.0

    # Dec in deg
    dec_deg = eph['DEC'][0]
```

```

# appending data to numpy arrays
data_ra_deg = numpy.append (data_ra_deg, ra_deg)
data_dec_deg = numpy.append (data_dec_deg, dec_deg)

# ecliptic plane
ecl_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
ecl_lat = numpy.zeros (1000) * u_deg
ecl_coord = astropy.coordinates.GeocentricMeanEcliptic (lon=ecl_lon, \
                                                         lat=ecl_lat, \
                                                         obstime=date)

ecl_ra = ecl_coord.transform_to (astropy.coordinates.ICRS) \
        .ra.wrap_at (180.0 * u_deg).radian
ecl_dec = ecl_coord.transform_to (astropy.coordinates.ICRS).dec.radian

# galactic plane
gal_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
gal_lat = numpy.zeros (1000) * u_deg
gal_coord = astropy.coordinates.Galactic (l=gal_lon, \
                                          b=gal_lat)

gal_ra = gal_coord.transform_to (astropy.coordinates.ICRS) \
        .ra.wrap_at (180.0 * u_deg).radian
gal_dec = gal_coord.transform_to (astropy.coordinates.ICRS).dec.radian

# making objects "fig" and "ax"
fig = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax = fig.add_subplot (111, projection='hammer')

# axes
ax.grid ()
ax.set_xlabel ('Right Ascension [deg]')
ax.set_ylabel ('Declination [deg]')

# title
text_title = f"Distribution of asteroids"
ax.set_title (text_title, loc='right')

# plotting data
ax.plot (ecl_ra, ecl_dec, \
        linestyle='None', marker='o', color='y', markersize=3, alpha=0.3, \
        label='Ecliptic plane')
ax.plot (ecl_ra, ecl_dec, \
        linestyle='None', marker='o', color='y', markersize=30, alpha=0.01)
ax.plot (gal_ra, gal_dec, \
        linestyle='None', marker='o', color='silver', markersize=5, \
        alpha=0.5, label='Galactic plane')
ax.plot (numpy.deg2rad (data_ra_deg), numpy.deg2rad (data_dec_deg), \
        linestyle='None', marker='o', color='b', markersize=3, alpha=0.3, \
        label='Asteroids')
ax.legend (bbox_to_anchor=(0.9, -0.1))

# saving file
fig.savefig (file_output, dpi=225)

```

Execute above script to make a plot of positions of 3000 asteroids on the sky on 01/Jan/2023 using Hammer projection.

```

% chmod a+x ai202209_s10_00_15.py
% ./ai202209_s10_00_15.py
1 Ceres (A801 AA) : (RA, Dec) = (187.3734 deg, +9.7846 deg)

```

```

2 Pallas (A802 FA)      : (RA, Dec) = (103.8890 deg, -32.0147 deg)
3 Juno (A804 RA)       : (RA, Dec) = (357.4604 deg, -8.5643 deg)
4 Vesta (A807 FA)      : (RA, Dec) = (347.4824 deg, -12.2712 deg)
5 Astraea (A845 XA)    : (RA, Dec) = (349.9594 deg, -8.1209 deg)
6 Hebe (A847 NA)       : (RA, Dec) = (132.4608 deg, +9.7006 deg)
7 Iris (A847 PA)       : (RA, Dec) = (212.8121 deg, -17.8497 deg)
8 Flora (A847 UA)      : (RA, Dec) = (267.6340 deg, -21.4327 deg)
9 Metis (A848 HA)      : (RA, Dec) = (325.6073 deg, -18.4135 deg)
10 Hygiea (A849 GA)    : (RA, Dec) = (275.1425 deg, -23.8649 deg)

...

2991 Bilbo (1982 HV)    : (RA, Dec) = ( 17.9072 deg, +2.2854 deg)
2992 Vondel (2540 P-L) : (RA, Dec) = (275.9027 deg, -17.8996 deg)
2993 Wendy (1970 PA)   : (RA, Dec) = (207.1615 deg, -22.0275 deg)
2994 Flynn (1975 PA)   : (RA, Dec) = (183.6456 deg, -0.8351 deg)
2995 Taratuta (1978 QK) : (RA, Dec) = (215.2461 deg, -8.1545 deg)
2996 Bowman (1954 RJ)  : (RA, Dec) = (258.2920 deg, -25.7335 deg)
2997 Cabrera (1974 MJ) : (RA, Dec) = (243.2244 deg, -26.0287 deg)
2998 Berendeya (1975 TR3): (RA, Dec) = (213.1368 deg, -11.4399 deg)
2999 Dante (1981 CY)   : (RA, Dec) = (248.6859 deg, -20.5187 deg)
3000 Leonardo (1981 EG19): (RA, Dec) = (354.7845 deg, -2.3956 deg)

```

Display the PNG file. (Fig. 16)

```
% feh -dF ai202209_s10_00_15.png
```

Try following practice.

Practice 10-16

Get positions of 10000 asteroids from asteroid number 1 to asteroid number 10000 on 01/Dec/2023 using **astroquery**, and make a plot of asteroid positions on the sky on 01/Dec/2023 using Aitoff projection. Superimpose the location of the Ecliptic plane and the Galactic plane on the plot.

2.13 Making a density plot

Make a Python script to fetch positions of 3000 asteroids using **astroquery**, and make a density plot of distribution of asteroids on the sky using Hammer projection. Here is an example.

Python Code 17: ai202209_s10_00_16.py

```

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

#
# Time-stamp: <2022/11/20 23:17:59 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.coordinates
import astropy.time
import astropy.units

# importing astroquery module
import astroquery.jplhorizons

```

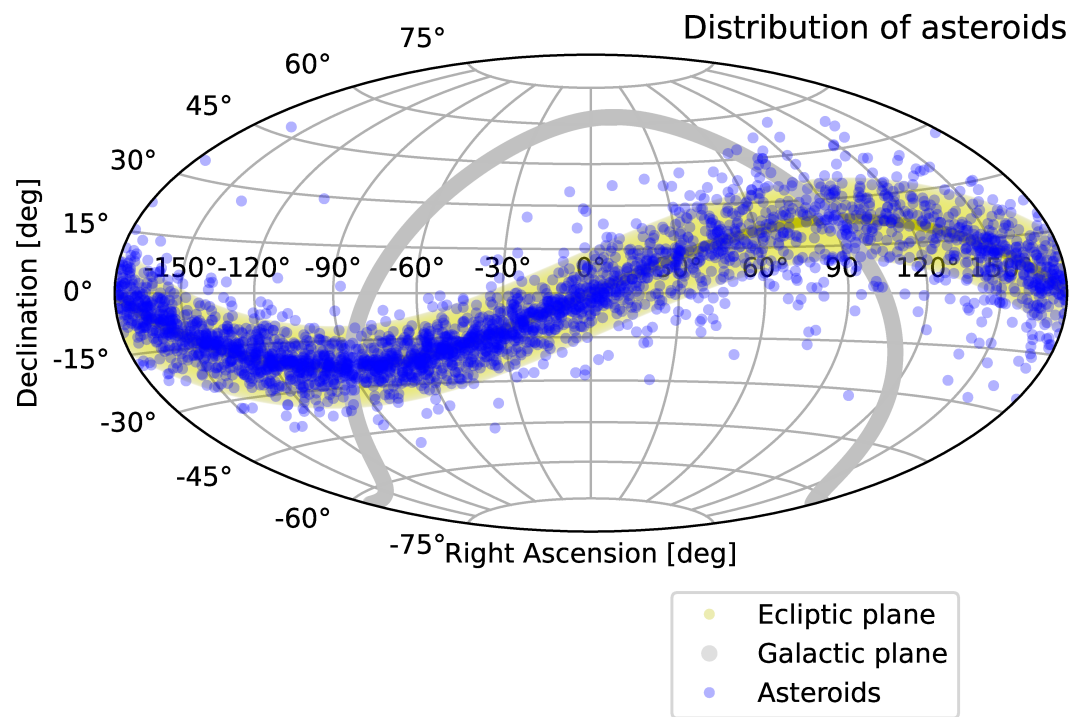


Figure 16: Positions of 3000 asteroids on the sky on 01/Jan/2023 using Hammer projection. The location of the Ecliptic plane and the Galactic plane are superimposed on the plot.

```

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

# output file name
file_output = 'ai202209_s10_00_16.png'

# units
u_deg = astropy.units.degree

# date/time
date = astropy.time.Time ('2023-01-01 00:00:00')

# number of asteroids to get position
n_asteroids = 3000

# making empty numpy arrays for storing data
data_ra_deg = numpy.array ([])
data_dec_deg = numpy.array ([])

# processing for each asteroid
for i in range (1, n_asteroids + 1):
    # set-up a query for JPL Horizons
    query = astroquery.jplhorizons.Horizons (id=f"{i}", \
                                              id_type='smallbody', \
                                              epochs=date.jd)

    # fetching ephemeris of asteroid
    eph = query.ephemerides ()

    # printing RA and Dec of asteroid
    print (f"{eph['targetname'][0]:24s}:" \
          + f" (RA, Dec) = ({eph['RA'][0]:8.4f} deg," \
          + f" {eph['DEC'][0]:+8.4f} deg)")

    # RA in deg
    ra_deg = eph['RA'][0]
    if (ra_deg > 180.0):
        ra_deg -= 360.0

    # Dec in deg
    dec_deg = eph['DEC'][0]

    # appending data to numpy arrays
    data_ra_deg = numpy.append (data_ra_deg, ra_deg)
    data_dec_deg = numpy.append (data_dec_deg, dec_deg)

# ecliptic plane
ecl_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
ecl_lat = numpy.zeros (1000) * u_deg
ecl_coord = astropy.coordinates.GeocentricMeanEcliptic (lon=ecl_lon, \
                                                         lat=ecl_lat, \
                                                         obstime=date)

ecl_ra = ecl_coord.transform_to (astropy.coordinates.ICRS) \
        .ra.wrap_at (180.0 * u_deg).radian
ecl_dec = ecl_coord.transform_to (astropy.coordinates.ICRS).dec.radian

# galactic plane

```

```

gal_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
gal_lat = numpy.zeros (1000) * u_deg
gal_coord = astropy.coordinates.Galactic (l=gal_lon, \
                                          b=gal_lat)
gal_ra = gal_coord.transform_to (astropy.coordinates.ICRS) \
        .ra.wrap_at (180.0 * u_deg).radian
gal_dec = gal_coord.transform_to (astropy.coordinates.ICRS).dec.radian

# making objects "fig" and "ax"
fig = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax = fig.add_subplot (111, projection='hammer')

# axes
ax.grid ()
ax.set_xlabel ('Right Ascension [deg]')
ax.set_ylabel ('Declination [deg]')

# title
text_title = f"Distribution of asteroids"
ax.set_title (text_title, loc='right')

# plotting data
density = ax.hexbin (numpy.deg2rad (data_ra_deg), numpy.deg2rad (data_dec_deg), \
                    gridsize=90, cmap=matplotlib.cm.plasma)
fig.colorbar (density, ax=ax, spacing='uniform', extend='both')

# saving file
fig.savefig (file_output, dpi=225)

```

Execute above script to make a density plot of positions of 3000 asteroids on the sky on 01/Jan/2023 using Hammer projection.

```

% chmod a+x ai202209_s10_00_16.py
% ./ai202209_s10_00_16.py
1 Ceres (A801 AA) : (RA, Dec) = (187.3734 deg, +9.7846 deg)
2 Pallas (A802 FA) : (RA, Dec) = (103.8890 deg, -32.0147 deg)
3 Juno (A804 RA) : (RA, Dec) = (357.4604 deg, -8.5643 deg)
4 Vesta (A807 FA) : (RA, Dec) = (347.4824 deg, -12.2712 deg)
5 Astraea (A845 XA) : (RA, Dec) = (349.9594 deg, -8.1209 deg)
6 Hebe (A847 NA) : (RA, Dec) = (132.4608 deg, +9.7006 deg)
7 Iris (A847 PA) : (RA, Dec) = (212.8121 deg, -17.8497 deg)
8 Flora (A847 UA) : (RA, Dec) = (267.6340 deg, -21.4327 deg)
9 Metis (A848 HA) : (RA, Dec) = (325.6073 deg, -18.4135 deg)
10 Hygiea (A849 GA) : (RA, Dec) = (275.1425 deg, -23.8649 deg)

...

2991 Bilbo (1982 HV) : (RA, Dec) = ( 17.9072 deg, +2.2854 deg)
2992 Vondel (2540 P-L) : (RA, Dec) = (275.9027 deg, -17.8996 deg)
2993 Wendy (1970 PA) : (RA, Dec) = (207.1615 deg, -22.0275 deg)
2994 Flynn (1975 PA) : (RA, Dec) = (183.6456 deg, -0.8351 deg)
2995 Taratuta (1978 QK) : (RA, Dec) = (215.2461 deg, -8.1545 deg)
2996 Bowman (1954 RJ) : (RA, Dec) = (258.2920 deg, -25.7335 deg)
2997 Cabrera (1974 MJ) : (RA, Dec) = (243.2244 deg, -26.0287 deg)
2998 Berendeya (1975 TR3) : (RA, Dec) = (213.1368 deg, -11.4399 deg)
2999 Dante (1981 CY) : (RA, Dec) = (248.6859 deg, -20.5187 deg)
3000 Leonardo (1981 EG19) : (RA, Dec) = (354.7845 deg, -2.3956 deg)

```

Display the PNG file. (Fig. 17)

```
% feh -dF ai202209_s10_00_16.png
```

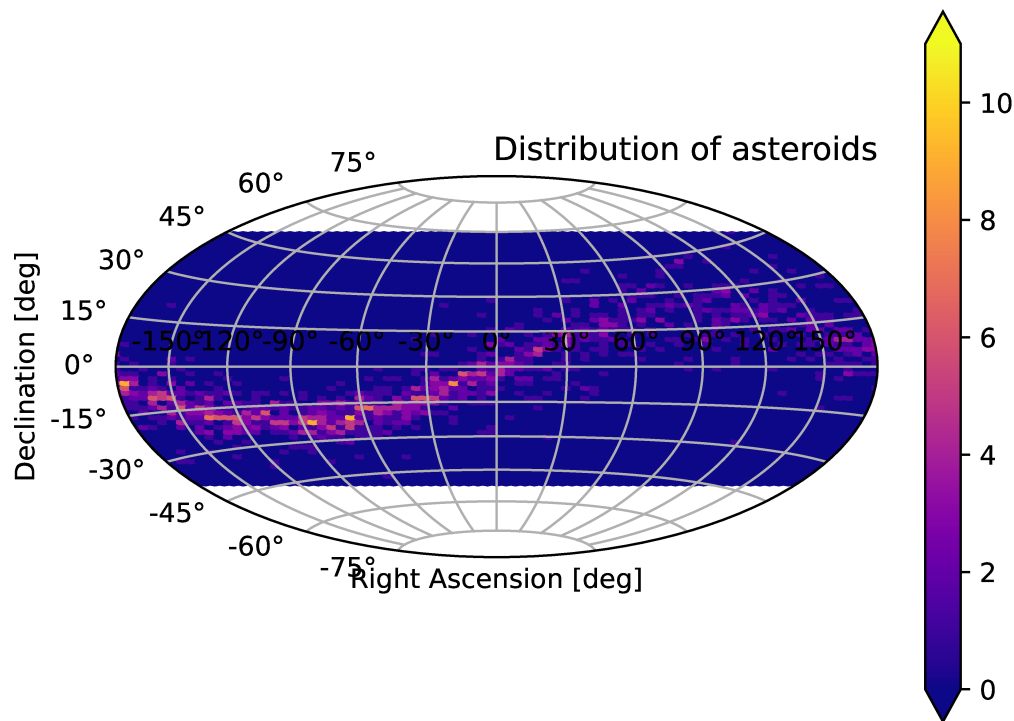


Figure 17: A density plot of 3000 asteroids on the sky on 01/Jan/2023 using Hammer projection.

Try following practice.

Practice 10-17

Get positions of 10000 asteroids from asteroid number 1 to asteroid number 10000 on 01/Dec/2023 using `astroquery`, and make a density plot of asteroids on the sky on 01/Dec/2023 using Mollweide projection.

3 Distribution of stars from Bright Star Catalogue

3.1 Downloading Bright Star Catalogue

Download Bright Star Catalogue. Here is a sample script.

Python Code 18: ai202209_s10_01_00.py

```
#!/usr/pkg/bin/python3.9
#
# Time-stamp: <2022/11/20 19:32:12 (CST) daisuke>
#
```

```

# importing urllib module
import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# URL of data file
url_data = 'https://cdsarc.cds.unistra.fr/ftp/V/50/catalog.gz'

# output file name
file_output = 'bsc5.cat.gz'

# printing status
print (f'Now, fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()

# printing status
print (f'Finished fetching {url_data}!')

# printing status
print (f'Now, writing the data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'wb') as fh_write:
    # writing data
    fh_write.write (data_byte)

# printing status
print (f'Finished writing the data into file "{file_output}"!')

```

Execute above script to download the Hipparchos catalogue.

```

% chmod a+x ai202209_s10_01_00.py
% ./ai202209_s10_01_00.py
Now, fetching https://cdsarc.cds.unistra.fr/ftp/V/50/catalog.gz...
Finished fetching https://cdsarc.cds.unistra.fr/ftp/V/50/catalog.gz!
Now, writing the data into file "bsc5.cat.gz"...
Finished writing the data into file "bsc5.cat.gz"!
% ls -lF bsc5.cat.gz
-rw-r--r--  1 daisuke  wheel  573921 Nov 20 19:32 bsc5.cat.gz

```

Also, download “ReadMe” file of Bright Star Catalogue.

Python Code 19: ai202209_s10_01_01.py

```

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

#
# Time-stamp: <2022/11/20 19:32:21 (CST) daisuke>
#

# importing urllib module

```

```

import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# URL of data file
url_data = 'http://cdsarc.u-strasbg.fr/ftp/V/50/ReadMe'

# output file name
file_output = 'bsc5.readme'

# printing status
print (f'Now, fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()

# printing status
print (f'Finished fetching {url_data}!')

# converting raw byte data into string
data_str = data_byte.decode ('utf-8')

# printing status
print (f'Now, writing data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'w') as fh_write:
    # writing data
    fh_write.write (data_str)

# printing status
print (f'Finished writing data into file "{file_output}"!')

```

Execute above script to download the Bright Star Catalogue.

```

% chmod a+x ai202209_s10_01_01.py
% ./ai202209_s10_01_01.py
Now, fetching http://cdsarc.u-strasbg.fr/ftp/V/50/ReadMe...
Finished fetching http://cdsarc.u-strasbg.fr/ftp/V/50/ReadMe!
Now, writing data into file "bsc5.readme"...
Finished writing data into file "bsc5.readme"!
% ls -lF bsc5.*
-rw-r--r--  1 daisuke  wheel   573921 Nov 20 19:32 bsc5.cat.gz
-rw-r--r--  1 daisuke  wheel    11571 Nov 20 19:34 bsc5.readme

```

3.2 Reading Bright Star Catalogue

Make a Python script to read Bright Star Catalogue. Here is an example.

Python Code 20: ai202209_s10_01_02.py

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

```
#
# Time-stamp: <2022/11/20 19:40:29 (CST) daisuke>
#

# importing gzip module
import gzip

# catalogue file
file_catalogue = 'bsc5.cat.gz'

# dictionary for storing data
bsc = {}

# opening file
with gzip.open (file_catalogue, 'rb') as fh:
    # reading the catalogue file line-by-line
    for line in fh:
        # decoding byte data
        line = line.decode ('utf-8')

        # finding values
        # Harvard Revised number
        hr_str = line[0:4].strip ()
        # RA_h
        RA_h_str = line[75:77].strip()
        # RA_m
        RA_m_str = line[77:79].strip()
        # RA_s
        RA_s_str = line[79:83].strip()
        # Dec_sign
        Dec_sign_str = line[83]
        # Dec_d
        Dec_d_str = line[84:86].strip ()
        # Dec_m
        Dec_m_str = line[86:88].strip ()
        # Dec_s
        Dec_s_str = line[88:90].strip ()
        # galactic longitude
        glon_str = line[90:96].strip ()
        # galactic latitude
        glat_str = line[96:102].strip ()
        # V-band magnitude
        Vmag_str = line[102:107].strip ()

        # skip processing, if any of necessary fields is blank.
        if ( (hr_str == '') or (RA_h_str == '') or (Dec_sign_str == '')
            or (glon_str == '') or (Vmag_str == '') ):
            continue

        # HR number
        hr = int (hr_str)
        # RA
        RA_h = int (RA_h_str)
        RA_m = int (RA_m_str)
        RA_s = float (RA_s_str)
        RA = "%02d:%02d:%04.1f" % (RA_h, RA_m, RA_s)
        # Dec
        Dec_sign = Dec_sign_str
        Dec_d = int (Dec_d_str)
```

```

Dec_m = int (Dec_m_str)
Dec_s = int (Dec_s_str)
Dec = "%s%02d:%02d:%02d" % (Dec_sign, Dec_d, Dec_m, Dec_s)
# galactic longitude
glon = float (glon_str)
# galactic latitude
glat = float (glat_str)
# V-band magnitude
Vmag = float (Vmag_str)

# making 2-dim. dictionary to store data
bsc[hr] = {}
bsc[hr]['RA'] = RA
bsc[hr]['Dec'] = Dec
bsc[hr]['glon'] = glon
bsc[hr]['glat'] = glat
bsc[hr]['Vmag'] = Vmag

# sorting data by V-band magnitude
for obj in sorted (bsc, key=lambda x: bsc[x]['Vmag']):
    # printing data
    print ("%4s %-10s %-9s %6.2f %6.2f %5.2f" \
            % (obj, bsc[obj]['RA'], bsc[obj]['Dec'], \
              bsc[obj]['glon'], bsc[obj]['glat'], bsc[obj]['Vmag']))

```

Execute above script to read the Bright Star Catalogue.

```

% chmod a+x ai202209_s10_01_02.py
% ./ai202209_s10_01_02.py > bsc5.txt
% head bsc5.txt
2491 06:45:08.9 -16:42:58 227.22 -8.88 -1.46
2326 06:23:57.1 -52:41:45 261.21 -25.29 -0.72
5340 14:15:39.7 +19:10:57 15.14 69.11 -0.04
5459 14:39:35.9 -60:50:07 315.78 -0.71 -0.01
7001 18:36:56.3 +38:47:01 67.44 19.24 0.03
1708 05:16:41.4 +45:59:53 162.58 4.57 0.08
1713 05:14:32.3 -08:12:06 209.24 -25.25 0.12
2943 07:39:18.1 +05:13:30 213.69 13.03 0.38
472 01:37:42.9 -57:14:12 290.84 -58.79 0.46
2061 05:55:10.3 +07:24:25 199.79 -8.96 0.50

```

Try following practice.

Practice 10-16

Make a Python script to read Bright Star Catalogue and print the Harvard revised number, name, RA, Dec, V-band magnitude, (B-V) colour index, and spectral type of stars.

3.3 Plotting stars in Bright Star Catalogue

Make a Python script to plot locations of stars in Bright Star Catalogue using Hammer projection. Here is an example.

Python Code 21: ai202209_s10_01_03.py

```

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

#
# Time-stamp: <2022/11/20 20:10:11 (CST) daisuke>
#

```

```
# importing numpy module
import numpy

# importing astropy module
import astropy.units
import astropy.coordinates

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

# units
u_ha = astropy.units.hourangle
u_deg = astropy.units.deg

# data file
file_data = 'bsc5.txt'

# output files
file_output = 'ai202209_s10_01_03.png'

# numpy arrays to storing data
data_hr = numpy.array ([])
data_ra = numpy.array ([])
data_dec = numpy.array ([])
data_l = numpy.array ([])
data_b = numpy.array ([])
data_Vmag = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # counting number of objects
    n = 0
    for line in fh:
        n += 1

# opening file
with open (file_data, 'r') as fh:
    # initialising counter
    i = 0
    # reading file
    for line in fh:
        # counting and showing progress
        i += 1
        if (i % 500 == 0):
            print ("progress: %4d / %4d" % (i, n) )
        # splitting line
        (hr_str, ra_str, dec_str, glon_str, glat_str, Vmag_str) = line.split ()
        # conversion from string to int or float
        hr = int (hr_str)
        glon_deg = float (glon_str)
        glat_deg = float (glat_str)
        Vmag = float (Vmag_str)

        # skip if Vmag > 6.0
        if (Vmag > 6.0):
            continue
```

```

# coordinate
coord = astropy.coordinates.SkyCoord (ra_str, dec_str, \
                                       frame=astropy.coordinates.FK5, \
                                       equinox="J2000", \
                                       unit=(u_ha, u_deg) )

# (RA, Dec) in radian
ra_rad = coord.ra.radian
dec_rad = coord.dec.radian
# conversion from (RA, Dec) to (l, b) using astropy
l_rad = coord.galactic.l.radian
b_rad = coord.galactic.b.radian

# changing from [0:2pi] to [-pi:pi]
if (ra_rad > numpy.pi):
    ra_rad -= 2.0 * numpy.pi
if (l_rad > numpy.pi):
    l_rad -= 2.0 * numpy.pi

# appending data to numpy arrays
data_hr = numpy.append (data_hr, hr)
data_ra = numpy.append (data_ra, ra_rad)
data_dec = numpy.append (data_dec, dec_rad)
data_l = numpy.append (data_l, l_rad)
data_b = numpy.append (data_b, b_rad)
data_Vmag = numpy.append (data_Vmag, Vmag)

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

# axes
ax.grid ()
ax.set_title ('Bright Star Catalogue', loc='right')
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# plotting data
size = (6.5 - data_Vmag) * 5.0
ax.scatter (data_ra, data_dec, marker='o', s=size, c='b', alpha=0.3)

# saving figure to files
fig.savefig (file_output, dpi=225, bbox_inches="tight")

```

Execute above script to plot locations of stars in Bright Star Catalogue using Hammer projection.

```

% chmod a+x ai202209_s10_01_03.py
% ./ai202209_s10_01_03.py
progress: 500 / 9096
progress: 1000 / 9096
progress: 1500 / 9096
progress: 2000 / 9096
progress: 2500 / 9096
progress: 3000 / 9096
progress: 3500 / 9096
progress: 4000 / 9096
progress: 4500 / 9096
progress: 5000 / 9096

```

```

progress: 5500 / 9096
progress: 6000 / 9096
progress: 6500 / 9096
progress: 7000 / 9096
progress: 7500 / 9096
progress: 8000 / 9096
progress: 8500 / 9096
progress: 9000 / 9096

```

Display the PNG file. (Fig. 18)

```
% feh -dF ai202209_s10_01_03.png
```

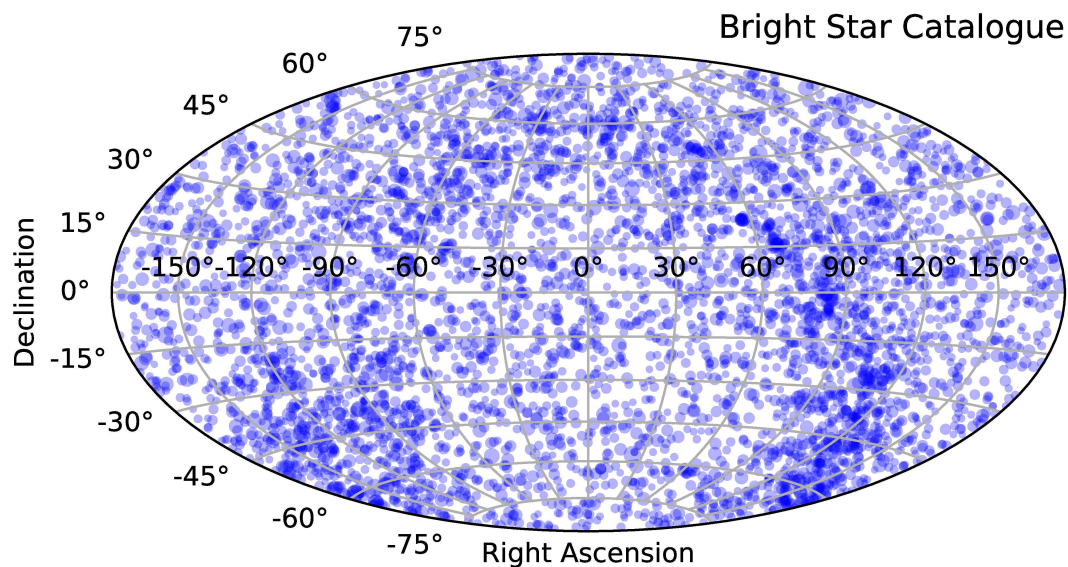


Figure 18: Positions of stars in Bright Star Catalogue using Hammer projection.

Try following practice.

Practice 10-17

Make a Python script to plot locations of stars in Bright Star Catalogue using Mollweide projection.

Make a Python script to plot locations of stars in Bright Star Catalogue using Hammer projection and superimpose the location of the Galactic plane. Here is an example.

Python Code 22: ai202209_s10_01_04.py

```

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

#
# Time-stamp: <2022/11/20 20:21:40 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.units

```

```
import astropy.coordinates

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

# units
u_ha = astropy.units.hourangle
u_deg = astropy.units.deg

# data file
file_data = 'bsc5.txt'

# output files
file_output = 'ai202209_s10_01_04.pdf'

# numpy arrays to storing data
data_hr = numpy.array ([])
data_ra = numpy.array ([])
data_dec = numpy.array ([])
data_l = numpy.array ([])
data_b = numpy.array ([])
data_Vmag = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # counting number of objects
    n = 0
    for line in fh:
        n += 1

# opening file
with open (file_data, 'r') as fh:
    # initialising counter
    i = 0
    # reading file
    for line in fh:
        # counting and showing progress
        i += 1
        if (i % 500 == 0):
            print ("progress: %4d / %4d" % (i, n) )
        # splitting line
        (hr_str, ra_str, dec_str, glon_str, glat_str, Vmag_str) = line.split ()
        # conversion from string to int or float
        hr = int (hr_str)
        glon_deg = float (glon_str)
        glat_deg = float (glat_str)
        Vmag = float (Vmag_str)

        # skip if Vmag > 6.0
        if (Vmag > 6.0):
            continue

        # coordinate
        coord = astropy.coordinates.SkyCoord (ra_str, dec_str, \
                                                frame=astropy.coordinates.FK5, \
                                                equinox="J2000", \
                                                unit=(u_ha, u_deg) )
```

```

# (RA, Dec) in radian
ra_rad = coord.ra.radian
dec_rad = coord.dec.radian
# conversion from (RA, Dec) to (l, b) using astropy
l_rad = coord.galactic.l.radian
b_rad = coord.galactic.b.radian

# changing from [0:2pi] to [-pi:pi]
if (ra_rad > numpy.pi):
    ra_rad -= 2.0 * numpy.pi
if (l_rad > numpy.pi):
    l_rad -= 2.0 * numpy.pi

# appending data to numpy arrays
data_hr = numpy.append (data_hr, hr)
data_ra = numpy.append (data_ra, ra_rad)
data_dec = numpy.append (data_dec, dec_rad)
data_l = numpy.append (data_l, l_rad)
data_b = numpy.append (data_b, b_rad)
data_Vmag = numpy.append (data_Vmag, Vmag)

# galactic plane
gal_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
gal_lat = numpy.zeros (1000) * u_deg
gal_coord = astropy.coordinates.Galactic (l=gal_lon, \
                                           b=gal_lat)
gal_ra = gal_coord.transform_to (astropy.coordinates.ICRS) \
        .ra.wrap_at (180.0 * u_deg).radian
gal_dec = gal_coord.transform_to (astropy.coordinates.ICRS).dec.radian

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

# axes
ax.grid ()
ax.set_title ('Bright Star Catalogue', loc='right')
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# plotting data
ax.plot (gal_ra, gal_dec, \
        linestyle='None', marker='o', color='silver', markersize=5, \
        alpha=0.1, label='Galactic plane')
size = (6.5 - data_Vmag) * 5.0
ax.scatter (data_ra, data_dec, marker='o', s=size, c='b', alpha=0.3)

# saving figure to files
fig.savefig (file_output, dpi=225, bbox_inches="tight")

```

Execute above script to plot locations of stars in Bright Star Catalogue and the Galactic plane using Hammer projection.

```

% chmod a+x ai202209_s10_01_04.py
% ./ai202209_s10_01_04.py
progress: 500 / 9096
progress: 1000 / 9096
progress: 1500 / 9096

```

```

progress: 2000 / 9096
progress: 2500 / 9096
progress: 3000 / 9096
progress: 3500 / 9096
progress: 4000 / 9096
progress: 4500 / 9096
progress: 5000 / 9096
progress: 5500 / 9096
progress: 6000 / 9096
progress: 6500 / 9096
progress: 7000 / 9096
progress: 7500 / 9096
progress: 8000 / 9096
progress: 8500 / 9096
progress: 9000 / 9096

```

Display the PNG file. (Fig. 19)

```
% feh -dF ai202209_s10_01_04.png
```

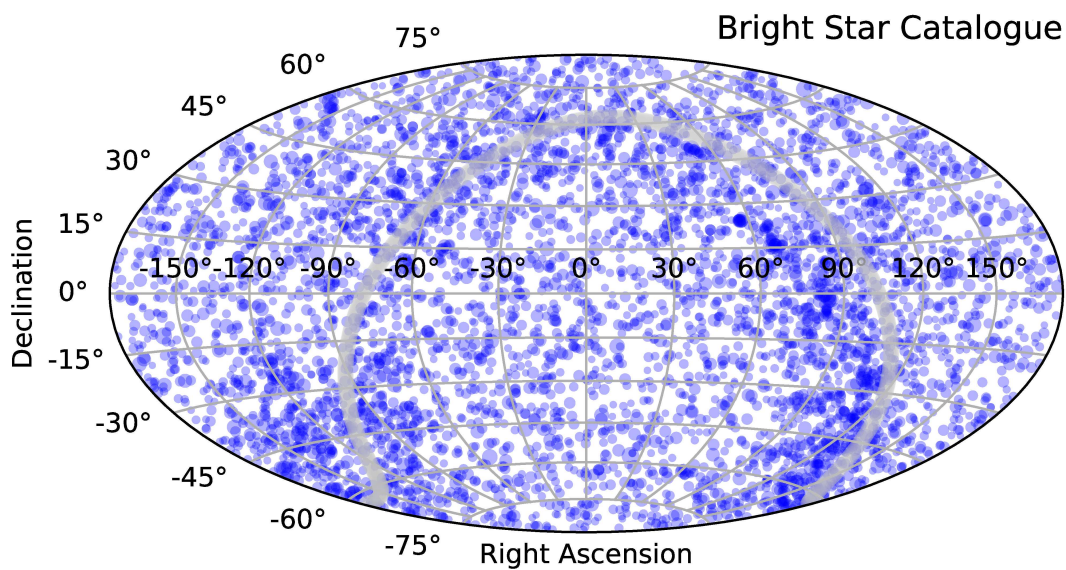


Figure 19: Positions of stars in Bright Star Catalogue using Hammer projection. The location of the Galactic plane is superimposed on the plot.

Try following practice.

Practice 10-18

Make a Python script to plot locations of stars in Bright Star Catalogue using Mollweide projection. Superimpose the location of the Galactic plane.

Make a Python script to plot locations of stars in Bright Star Catalogue on galactic coordinate using Hammer projection. Here is an example.

Python Code 23: ai202209_s10_01_05.py

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

```
#
# Time-stamp: <2022/11/20 20:28:20 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.units
import astropy.coordinates

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

# units
u_ha = astropy.units.hourangle
u_deg = astropy.units.deg

# data file
file_data = 'bsc5.txt'

# output files
file_output = 'ai202209_s10_01_05.png'

# numpy arrays to storing data
data_hr = numpy.array ([])
data_ra = numpy.array ([])
data_dec = numpy.array ([])
data_l = numpy.array ([])
data_b = numpy.array ([])
data_Vmag = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # counting number of objects
    n = 0
    for line in fh:
        n += 1

# opening file
with open (file_data, 'r') as fh:
    # initialising counter
    i = 0
    # reading file
    for line in fh:
        # counting and showing progress
        i += 1
        if (i % 500 == 0):
            print ("progress: %4d / %4d" % (i, n) )
        # splitting line
        (hr_str, ra_str, dec_str, glon_str, glat_str, Vmag_str) = line.split ()
        # conversion from string to int or float
        hr = int (hr_str)
        glon_deg = float (glon_str)
        glat_deg = float (glat_str)
        Vmag = float (Vmag_str)

        # skip if Vmag > 6.0
```

```

    if (Vmag > 6.0):
        continue

    # coordinate
    coord = astropy.coordinates.SkyCoord (ra_str, dec_str, \
                                           frame=astropy.coordinates.FK5, \
                                           equinox="J2000", \
                                           unit=(u_ha, u_deg) )

    # (RA, Dec) in radian
    ra_rad = coord.ra.radian
    dec_rad = coord.dec.radian
    # conversion from (RA, Dec) to (l, b) using astropy
    l_rad = coord.galactic.l.radian
    b_rad = coord.galactic.b.radian

    # changing from [0:2pi] to [-pi:pi]
    if (ra_rad > numpy.pi):
        ra_rad -= 2.0 * numpy.pi
    if (l_rad > numpy.pi):
        l_rad -= 2.0 * numpy.pi

    # appending data to numpy arrays
    data_hr = numpy.append (data_hr, hr)
    data_ra = numpy.append (data_ra, ra_rad)
    data_dec = numpy.append (data_dec, dec_rad)
    data_l = numpy.append (data_l, l_rad)
    data_b = numpy.append (data_b, b_rad)
    data_Vmag = numpy.append (data_Vmag, Vmag)

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

# axes
ax.grid ()
ax.set_title ('Bright Star Catalogue', loc='right')
ax.set_xlabel ('Galactic longitude')
ax.set_ylabel ('Galactic latitude')

# plotting data
size = (6.5 - data_Vmag) * 5.0
ax.scatter (data_l, data_b, marker='o', s=size, c='b', alpha=0.3)

# saving figure to files
fig.savefig (file_output, dpi=225, bbox_inches="tight")

```

Execute above script to plot locations of stars in Bright Star Catalogue on the galactic coordinate using Hammer projection.

```

% chmod a+x ai202209_s10_01_05.py
% ./ai202209_s10_01_05.py
progress: 500 / 9096
progress: 1000 / 9096
progress: 1500 / 9096
progress: 2000 / 9096
progress: 2500 / 9096
progress: 3000 / 9096

```

```

progress: 3500 / 9096
progress: 4000 / 9096
progress: 4500 / 9096
progress: 5000 / 9096
progress: 5500 / 9096
progress: 6000 / 9096
progress: 6500 / 9096
progress: 7000 / 9096
progress: 7500 / 9096
progress: 8000 / 9096
progress: 8500 / 9096
progress: 9000 / 9096

```

Display the PNG file. (Fig. 20)

```
% feh -dF ai202209_s10_01_05.png
```

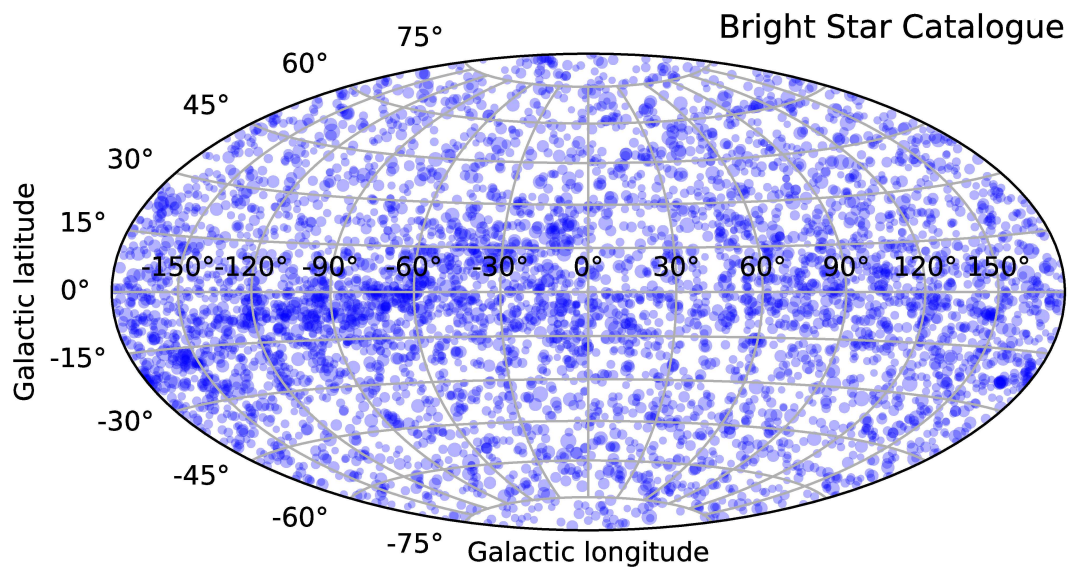


Figure 20: Positions of stars in Bright Star Catalogue on the galactic coordinate using Hammer projection.

Try following practice.

Practice 10-19

Make a Python script to plot locations of stars in Bright Star Catalogue on the galactic coordinate using Aitoff projection.

3.4 Extracting colour index from Bright Star Catalogue

Make a Python script to extract colour index from Bright Star Catalogue. Here is an example.

Python Code 24: ai202209_s10_01_06.py

```

#!/usr/pkg/bin/python3.9
#
# Time-stamp: <2022/11/20 20:36:04 (CST) daisuke>
#

```

```
# importing gzip module
import gzip

# catalogue file
file_catalogue = 'bsc5.cat.gz'

# dictionary for storing data
bsc = {}

# opening file
with gzip.open (file_catalogue, 'rb') as fh:
    # reading the catalogue file line-by-line
    for line in fh:
        # decoding byte data
        line = line.decode ('utf-8')

        # finding values
        # Harvard Revised number
        hr_str = line[0:4].strip ()
        # RA_h
        RA_h_str = line[75:77].strip()
        # RA_m
        RA_m_str = line[77:79].strip()
        # RA_s
        RA_s_str = line[79:83].strip()
        # Dec_sign
        Dec_sign_str = line[83]
        # Dec_d
        Dec_d_str = line[84:86].strip ()
        # Dec_m
        Dec_m_str = line[86:88].strip ()
        # Dec_s
        Dec_s_str = line[88:90].strip ()
        # galactic longitude
        glon_str = line[90:96].strip ()
        # galactic latitude
        glat_str = line[96:102].strip ()
        # V-band magnitude
        Vmag_str = line[102:107].strip ()
        # B-V colour index
        BV_colour_str = line[109:114].strip ()

        # skip processing, if any of necessary fields is blank.
        if ( (hr_str == '') or (RA_h_str == '') or (Dec_sign_str == '')
            or (glon_str == '') or (Vmag_str == '') or (BV_colour_str == '' ) ):
            continue

        # HR number
        hr = int (hr_str)
        # RA
        RA_h = int (RA_h_str)
        RA_m = int (RA_m_str)
        RA_s = float (RA_s_str)
        RA = "%02d:%02d:%04.1f" % (RA_h, RA_m, RA_s)
        # Dec
        Dec_sign = Dec_sign_str
        Dec_d = int (Dec_d_str)
        Dec_m = int (Dec_m_str)
```

```

Dec_s = int (Dec_s_str)
Dec = "%s%02d:%02d:%02d" % (Dec_sign, Dec_d, Dec_m, Dec_s)
# galactic longitude
glon = float (glon_str)
# galactic latitude
glat = float (glat_str)
# V-band magnitude
Vmag = float (Vmag_str)
# B-V colour index
BV_colour = float (BV_colour_str)

# making 2-dim. dictionary to store data
bsc[hr] = {}
bsc[hr]['RA'] = RA
bsc[hr]['Dec'] = Dec
bsc[hr]['glon'] = glon
bsc[hr]['glat'] = glat
bsc[hr]['Vmag'] = Vmag
bsc[hr]['BV'] = BV_colour

# printing header
print ("# star ID, RA, Dec, glon, glat, Vmag, B-V colour index")

# sorting data by V-band magnitude
for obj in sorted (bsc, key=lambda x: bsc[x]['Vmag']):
    # printing data
    print ("%4s %-10s %-9s %6.2f %6.2f %5.2f %5.2f" \
          % (obj, bsc[obj]['RA'], bsc[obj]['Dec'], \
            bsc[obj]['glon'], bsc[obj]['glat'], \
            bsc[obj]['Vmag'], bsc[obj]['BV']))

```

Execute above script to extract colour index from Bright Star Catalogue.

```

% chmod a+x ai202209_s10_01_06.py
% ./ai202209_s10_01_06.py > bsc5c.txt
% head bsc5c.txt
# star ID, RA, Dec, glon, glat, Vmag, B-V colour index
2491 06:45:08.9 -16:42:58 227.22 -8.88 -1.46 0.00
2326 06:23:57.1 -52:41:45 261.21 -25.29 -0.72 0.15
5340 14:15:39.7 +19:10:57 15.14 69.11 -0.04 1.23
5459 14:39:35.9 -60:50:07 315.78 -0.71 -0.01 0.71
7001 18:36:56.3 +38:47:01 67.44 19.24 0.03 0.00
1708 05:16:41.4 +45:59:53 162.58 4.57 0.08 0.80
1713 05:14:32.3 -08:12:06 209.24 -25.25 0.12 -0.03
2943 07:39:18.1 +05:13:30 213.69 13.03 0.38 0.42
472 01:37:42.9 -57:14:12 290.84 -58.79 0.46 -0.16

```

3.5 Plotting stars using different colour for different colour indices

Make a Python script to plot positions of stars in Bright Star Catalogue using different colour for different colour indices on the galactic coordinate. Here is an example.

Python Code 25: ai202209_s10_01_07.py

```

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

#
# Time-stamp: <2022/11/20 21:07:06 (CST) daisuke>
#

```

```
# importing numpy module
import numpy

# importing astropy module
import astropy.units
import astropy.coordinates

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

# units
u_ha = astropy.units.hourangle
u_deg = astropy.units.deg

# data file
file_data = 'bsc5c.txt'

# output files
file_output = 'ai202209_s10_01_07.png'

# numpy arrays to storing data
data_hr = numpy.array ([])
data_ra = numpy.array ([])
data_dec = numpy.array ([])
data_l = numpy.array ([])
data_b = numpy.array ([])
data_Vmag = numpy.array ([])
data_BV = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # counting number of objects
    n = 0
    for line in fh:
        n += 1

# opening file
with open (file_data, 'r') as fh:
    i = 0
    # reading file
    for line in fh:
        # skipping line, if the line starts with '#'
        if (line[0] == '#'):
            continue
        # counting and showing progress
        i += 1
        if (i % 500 == 0):
            print ("progress: %4d / %4d" % (i, n) )
        # splitting line
        (hr_str, ra_str, dec_str, glon_str, glat_str, Vmag_str, BV_str) \
            = line.split ()
        # conversion from string to int or float
        hr = int (hr_str)
        glon_deg = float (glon_str)
        glat_deg = float (glat_str)
        Vmag = float (Vmag_str)
        BV = float (BV_str)
```

```

# coordinate
coord = astropy.coordinates.SkyCoord (ra_str, dec_str, \
                                       frame=astropy.coordinates.FK5, \
                                       equinox="J2000", \
                                       unit=(u_ha, u_deg) )

# (RA, Dec) in radian
ra_rad = coord.ra.radian
dec_rad = coord.dec.radian
# conversion from (RA, Dec) to (l, b) using astropy
l_rad = coord.galactic.l.radian
b_rad = coord.galactic.b.radian

# changing from [0:2pi] to [-pi:pi]
if (ra_rad > numpy.pi):
    ra_rad -= 2.0 * numpy.pi
if (l_rad > numpy.pi):
    l_rad -= 2.0 * numpy.pi

# appending data to numpy arrays
data_hr = numpy.append (data_hr, hr)
data_ra = numpy.append (data_ra, ra_rad)
data_dec = numpy.append (data_dec, dec_rad)
data_l = numpy.append (data_l, l_rad)
data_b = numpy.append (data_b, b_rad)
data_Vmag = numpy.append (data_Vmag, Vmag)
data_BV = numpy.append (data_BV, BV)

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

# axes
ax.grid ()
ax.set_title ('Bright Star Catalogue', loc='right')
ax.set_xlabel ('Galactic longitude')
ax.set_ylabel ('Galactic latitude')

# plotting data
size = (8.0 - data_Vmag) * 5.0
colour = 2.0 - data_BV
for j in range ( len (colour) ):
    if (colour[j] < 0.0):
        colour[j] = 0.0
ax.scatter (data_l, data_b, s=size, c=colour, cmap=matplotlib.cm.Spectral, \
           marker='o', alpha=0.25)

# saving figure to files
fig.savefig (file_output, dpi=225, bbox_inches="tight")

```

Execute above script to plot positions of stars in Bright Star Catalogue using different colour for different colour indices.

```

% chmod a+x ai202209_s10_01_07.py
% ./ai202209_s10_01_07.py
progress: 500 / 8787
progress: 1000 / 8787

```

```

progress: 1500 / 8787
progress: 2000 / 8787
progress: 2500 / 8787
progress: 3000 / 8787
progress: 3500 / 8787
progress: 4000 / 8787
progress: 4500 / 8787
progress: 5000 / 8787
progress: 5500 / 8787
progress: 6000 / 8787
progress: 6500 / 8787
progress: 7000 / 8787
progress: 7500 / 8787
progress: 8000 / 8787
progress: 8500 / 8787

```

Display the PNG file. (Fig. 21) More blue stars are found near the Galactic plane.

```
% feh -dF ai202209_s10_01_07.png
```

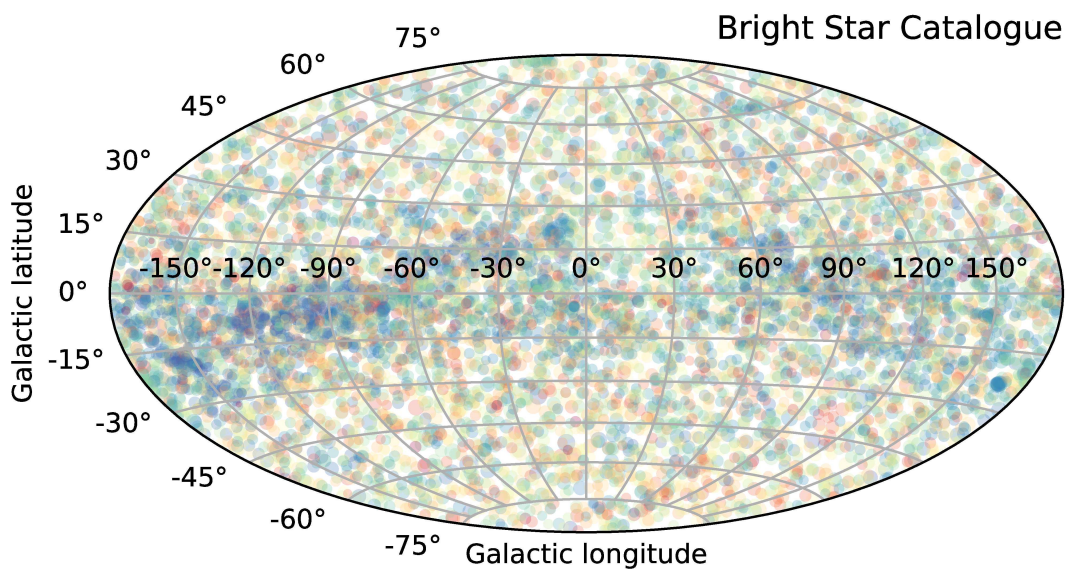


Figure 21: Positions of stars in Bright Star Catalogue on the galactic coordinate using Hammer projection.

Try following practice.

Practice 10-20

Make a Python script to plot locations of stars in Bright Star Catalogue on the galactic coordinate using Aitoff projection.

Make a Python script to plot positions of stars in Bright Star Catalogue using different colour for different colour indices on the equatorial coordinate. Here is an example.

Python Code 26: ai202209_s10_01_08.py

```

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

```

```
# Time-stamp: <2022/11/20 21:17:38 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.units
import astropy.coordinates

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

# units
u_ha = astropy.units.hourangle
u_deg = astropy.units.deg

# data file
file_data = 'bsc5c.txt'

# output files
file_output = 'ai202209_s10_01_08.png'

# numpy arrays to storing data
data_hr = numpy.array ([])
data_ra = numpy.array ([])
data_dec = numpy.array ([])
data_l = numpy.array ([])
data_b = numpy.array ([])
data_Vmag = numpy.array ([])
data_BV = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # counting number of objects
    n = 0
    for line in fh:
        n += 1

# opening file
with open (file_data, 'r') as fh:
    i = 0
    # reading file
    for line in fh:
        # skipping line, if the line starts with '#'
        if (line[0] == '#'):
            continue
        # counting and showing progress
        i += 1
        if (i % 500 == 0):
            print ("progress: %4d / %4d" % (i, n) )
        # splitting line
        (hr_str, ra_str, dec_str, glon_str, glat_str, Vmag_str, BV_str) \
            = line.split ()
        # conversion from string to int or float
        hr = int (hr_str)
        glon_deg = float (glon_str)
        glat_deg = float (glat_str)
```

```

Vmag = float (Vmag_str)
BV = float (BV_str)

# coordinate
coord = astropy.coordinates.SkyCoord (ra_str, dec_str, \
                                       frame=astropy.coordinates.FK5, \
                                       equinox="J2000", \
                                       unit=(u_ha, u_deg) )

# (RA, Dec) in radian
ra_rad = coord.ra.radian
dec_rad = coord.dec.radian
# conversion from (RA, Dec) to (l, b) using astropy
l_rad = coord.galactic.l.radian
b_rad = coord.galactic.b.radian

# changing from [0:2pi] to [-pi:pi]
if (ra_rad > numpy.pi):
    ra_rad -= 2.0 * numpy.pi
if (l_rad > numpy.pi):
    l_rad -= 2.0 * numpy.pi

# appending data to numpy arrays
data_hr = numpy.append (data_hr, hr)
data_ra = numpy.append (data_ra, ra_rad)
data_dec = numpy.append (data_dec, dec_rad)
data_l = numpy.append (data_l, l_rad)
data_b = numpy.append (data_b, b_rad)
data_Vmag = numpy.append (data_Vmag, Vmag)
data_BV = numpy.append (data_BV, BV)

# galactic plane
gal_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
gal_lat = numpy.zeros (1000) * u_deg
gal_coord = astropy.coordinates.Galactic (l=gal_lon, \
                                          b=gal_lat)

gal_ra = gal_coord.transform_to (astropy.coordinates.ICRS) \
        .ra.wrap_at (180.0 * u_deg).radian
gal_dec = gal_coord.transform_to (astropy.coordinates.ICRS).dec.radian

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

# axes
ax.grid ()
ax.set_title ('Bright Star Catalogue', loc='right')
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# plotting data
ax.plot (gal_ra, gal_dec, \
        linestyle='None', marker='o', color='silver', markersize=5, \
        alpha=0.1, label='Galactic plane')
size = (8.0 - data_Vmag) * 5.0
colour = 2.0 - data_BV
for j in range (len (colour) ):
    if (colour[j] < 0.0):

```

```

    colour[j] = 0.0
ax.scatter (data_ra, data_dec, s=size, c=colour, cmap=matplotlib.cm.Spectral, \
            marker='o', alpha=0.25)

# saving figure to files
fig.savefig (file_output, dpi=225, bbox_inches="tight")

```

Execute above script to plot positions of stars in Bright Star Catalogue using different colour for different colour indices on the equatorial coordinate.

```

% chmod a+x ai202209_s10_01_08.py
% ./ai202209_s10_01_08.py
progress: 500 / 8787
progress: 1000 / 8787
progress: 1500 / 8787
progress: 2000 / 8787
progress: 2500 / 8787
progress: 3000 / 8787
progress: 3500 / 8787
progress: 4000 / 8787
progress: 4500 / 8787
progress: 5000 / 8787
progress: 5500 / 8787
progress: 6000 / 8787
progress: 6500 / 8787
progress: 7000 / 8787
progress: 7500 / 8787
progress: 8000 / 8787
progress: 8500 / 8787

```

Display the PNG file. (Fig. 23)

```
% feh -dF ai202209_s10_01_08.png
```

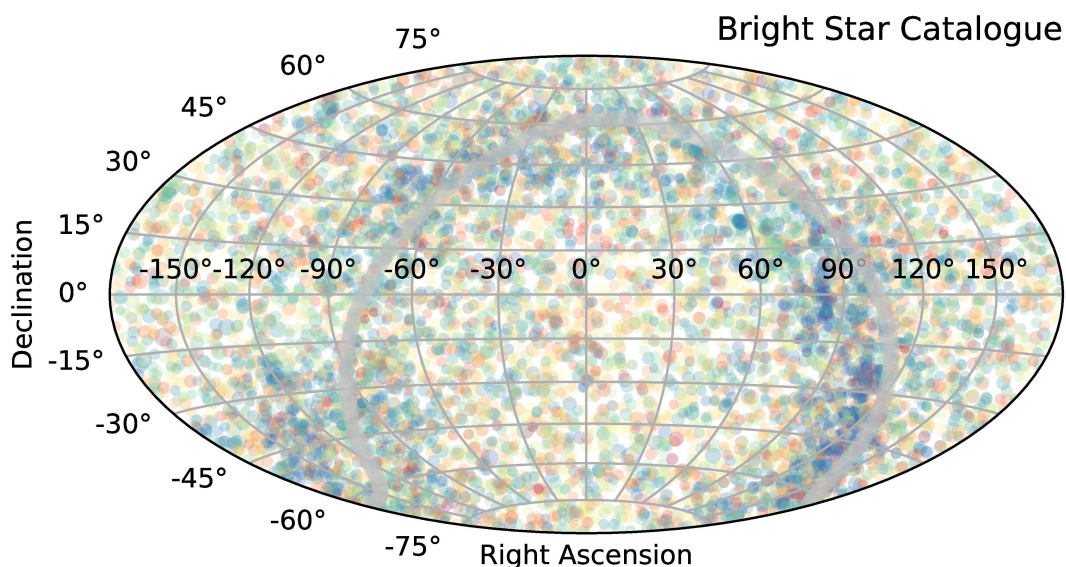


Figure 22: Positions of stars in Bright Star Catalogue on the equatorial coordinate using Hammer projection.

Try following practice.

Practice 10-21

Make a Python script to plot locations of stars in Bright Star Catalogue on the equatorial coordinate using Aitoff projection.

4 Distribution of stars from Hipparcos Catalogue

4.1 Downloading Hipparcos Catalogue

Make a Python script to download Hipparcos Catalogue. Here is an example.

Python Code 27: ai202209_s10_02_00.py

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

#
# Time-stamp: <2022/11/20 21:30:14 (CST) daisuke>
#

# importing urllib module
import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# URL of data file
url_data = 'https://cdsarc.cds.unistra.fr/ftp/I/311/hip2.dat.gz'

# output file name
file_output = 'hip2.dat.gz'

# printing status
print (f'Now, fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()

# printing status
print (f'Finished fetching {url_data}!')

# printing status
print (f'Now, writing the data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'wb') as fh_write:
    # writing data
    fh_write.write (data_byte)

# printing status
print (f'Finished writing the data into file "{file_output}"!')
```

Execute above script to download Hipparcos Catalogue.

```
% chmod a+x ai202209_s10_02_00.py
% ./ai202209_s10_02_00.py
Now, fetching https://cdsarc.cds.unistra.fr/ftp/I/311/hip2.dat.gz...
Finished fetching https://cdsarc.cds.unistra.fr/ftp/I/311/hip2.dat.gz!
Now, writing the data into file "hip2.dat.gz"...
Finished writing the data into file "hip2.dat.gz"!
```

4.2 Plotting locations of stars in Hipparcos Catalogue

Make a Python script to plot locations of stars in Hipparcos Catalogue. Here is an example.

Python Code 28: ai202209_s10_02_01.py

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

#
# Time-stamp: <2022/11/20 21:53:57 (CST) daisuke>
#

# importing gzip module
import gzip

# importing numpy module
import numpy

# importing astropy module
import astropy.units
import astropy.coordinates

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

# units
u_ha = astropy.units.hourangle
u_deg = astropy.units.deg
u_rad = astropy.units.rad

# files
file_cat = 'hip2.dat.gz'
file_output = 'ai202209_s10_02_01.png'

# list to store data
list_ra_rad = []
list_dec_rad = []

# counting number of data
n = 0

# opening file
with gzip.open(file_cat, 'rb') as fh:
    # reading data
    for line in fh:
        n += 1

# opening file
with gzip.open(file_cat, 'rb') as fh:
    # reading data
    i = 0
```

```

for line in fh:
    # decoding raw byte data
    line = line.decode ('utf-8')

    # extracting data
    ra_str  = line[15:28].strip ()
    dec_str = line[29:42].strip ()
    mag_str = line[129:136].strip ()
    BV_str  = line[152:158].strip ()

    # skipping, if any of data is missing.
    if ( (ra_str == '') or (dec_str == '') or (mag_str == '') \
        or (BV_str == '') ):
        continue

    # conversion from string to float
    ra_rad  = float (ra_str)
    dec_rad = float (dec_str)
    mag     = float (mag_str)
    BV      = float (BV_str)

    # coordinate
    coord = astropy.coordinates.SkyCoord (ra_rad, dec_rad, \
                                           frame=astropy.coordinates.ICRS, \
                                           unit=u_rad)

    # appending data to lists
    ra_rad_wrap = coord.ra.wrap_at (180 * u_deg).radian
    list_ra_rad.append (ra_rad_wrap)
    list_dec_rad.append (coord.dec.radian)

    # progress
    i += 1
    if (i % 5000 == 0):
        print ("progress: %6d / %6d" % (i, n) )

# making numpy arrays
data_ra_rad = numpy.array (list_ra_rad)
list_ra_rad.clear ()
data_dec_rad = numpy.array (list_dec_rad)
list_dec_rad.clear ()

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

matplotlib.rcParams['agg.path.chunksize'] = 10000

# axes
ax.grid ()
ax.set_title ('Hipparcos Catalogue', loc='right')
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# plotting data
ax.plot (data_ra_rad, data_dec_rad, marker='.', color='blue', \
        markersize=1, alpha=0.1, linestyle='None', fillstyle='full')

# saving file

```

```
fig.savefig (file_output , bbox_inches="tight" , dpi=225)
```

Execute above script to plot locations of stars in Hipparcos Catalogue.

```
% chmod a+x ai202209_s10_02_01.py
% ./ai202209_s10_02_01.py
progress: 5000 / 117955
progress: 10000 / 117955
progress: 15000 / 117955
progress: 20000 / 117955
progress: 25000 / 117955
progress: 30000 / 117955
progress: 35000 / 117955
progress: 40000 / 117955
progress: 45000 / 117955
progress: 50000 / 117955
progress: 55000 / 117955
progress: 60000 / 117955
progress: 65000 / 117955
progress: 70000 / 117955
progress: 75000 / 117955
progress: 80000 / 117955
progress: 85000 / 117955
progress: 90000 / 117955
progress: 95000 / 117955
progress: 100000 / 117955
progress: 105000 / 117955
progress: 110000 / 117955
progress: 115000 / 117955
```

Display the PNG file. (Fig. ??)

```
% feh -dF ai202209_s10_02_01.png
```

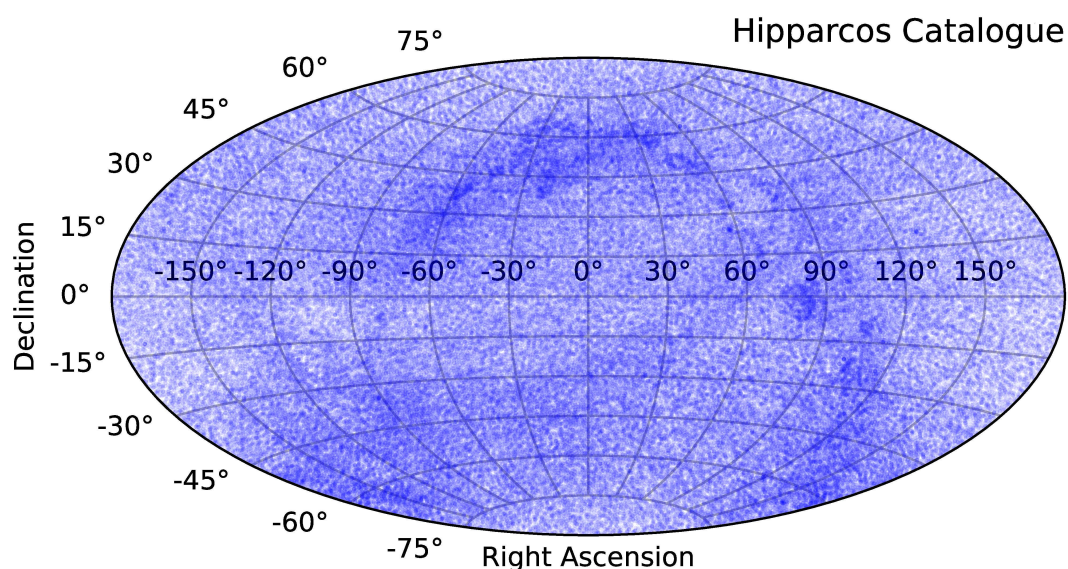


Figure 23: Positions of stars in Hipparcos Catalogue on the equatorial coordinate using Aitoff projection.

Try following practice.

Practice 10-22

Make a Python script to plot locations of stars in Hipparcos Catalogue on the equatorial coordinate using Mollweide projection.

Make a Python script to plot locations of stars in Hipparcos Catalogue and superimpose the location of the Galactic plane. Here is an example.

Python Code 29: ai202209_s10_02_02.py

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

#
# Time-stamp: <2022/11/20 22:07:40 (CST) daisuke>
#

# importing gzip module
import gzip

# importing numpy module
import numpy

# importing astropy module
import astropy.units
import astropy.coordinates

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

# units
u_ha = astropy.units.hourangle
u_deg = astropy.units.deg
u_rad = astropy.units.rad

# files
file_cat = 'hip2.dat.gz'
file_output = 'ai202209_s10_02_02.png'

# list to store data
list_ra_rad = []
list_dec_rad = []

# counting number of data
n = 0

# opening file
with gzip.open(file_cat, 'rb') as fh:
    # reading data
    for line in fh:
        n += 1

# opening file
with gzip.open(file_cat, 'rb') as fh:
    # reading data
    i = 0
    for line in fh:
        # decoding raw byte data
        line = line.decode('utf-8')
```

```

# extracting data
ra_str = line[15:28].strip ()
dec_str = line[29:42].strip ()
mag_str = line[129:136].strip ()
BV_str = line[152:158].strip ()

# skipping, if any of data is missing.
if ( (ra_str == '') or (dec_str == '') or (mag_str == '') \
    or (BV_str == '') ):
    continue
# conversion from string to float
ra_rad = float (ra_str)
dec_rad = float (dec_str)
mag     = float (mag_str)
BV      = float (BV_str)

# coordinate
coord = astropy.coordinates.SkyCoord (ra_rad, dec_rad, \
                                       frame=astropy.coordinates.ICRS, \
                                       unit=u_rad)

# appending data to lists
ra_rad_wrap = coord.ra.wrap_at (180 * u_deg).radian
list_ra_rad.append (ra_rad_wrap)
list_dec_rad.append (coord.dec.radian)

# progress
i += 1
if (i % 5000 == 0):
    print ("progress: %6d / %6d" % (i, n) )

# making numpy arrays
data_ra_rad = numpy.array (list_ra_rad)
list_ra_rad.clear ()
data_dec_rad = numpy.array (list_dec_rad)
list_dec_rad.clear ()

# galactic plane
gal_lon = numpy.linspace (0.001, 359.999, 1000) * u_deg
gal_lat = numpy.zeros (1000) * u_deg
gal_coord = astropy.coordinates.Galactic (l=gal_lon, \
                                          b=gal_lat)
gal_ra = gal_coord.transform_to (astropy.coordinates.ICRS) \
        .ra.wrap_at (180.0 * u_deg).radian
gal_dec = gal_coord.transform_to (astropy.coordinates.ICRS).dec.radian

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

matplotlib.rcParams['agg.path.chunksize'] = 10000

# axes
ax.grid ()
ax.set_title ('Hipparcos Catalogue', loc='right')
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

```

```
# plotting data
ax.plot (gal_ra, gal_dec, \
         linestyle='None', marker='o', color='magenta', markersize=5, \
         alpha=0.1, label='Galactic plane')
ax.plot (data_ra_rad, data_dec_rad, marker='.', color='blue', \
         markersize=1, alpha=0.1, linestyle='None', fillstyle='full')

# saving file
fig.savefig (file_output, bbox_inches="tight", dpi=225)
```

Execute above script to plot locations of stars in Hipparcos Catalogue.

```
% chmod a+x ai202209_s10_02_02.py
% ./ai202209_s10_02_02.py
progress: 5000 / 117955
progress: 10000 / 117955
progress: 15000 / 117955
progress: 20000 / 117955
progress: 25000 / 117955
progress: 30000 / 117955
progress: 35000 / 117955
progress: 40000 / 117955
progress: 45000 / 117955
progress: 50000 / 117955
progress: 55000 / 117955
progress: 60000 / 117955
progress: 65000 / 117955
progress: 70000 / 117955
progress: 75000 / 117955
progress: 80000 / 117955
progress: 85000 / 117955
progress: 90000 / 117955
progress: 95000 / 117955
progress: 100000 / 117955
progress: 105000 / 117955
progress: 110000 / 117955
progress: 115000 / 117955
```

Display the PNG file. (Fig. 24)

```
% feh -dF ai202209_s10_02_02.png
```

Try following practice.

Practice 10-23

Make a Python script to plot locations of stars in Hipparcos Catalogue and the Galactic plane on the equatorial coordinate using Mollweide projection.

4.3 Making a density plot

Make a Python script to make a density plot of stars in Hipparcos Catalogue. Here is an example.

Python Code 30: ai202209_s10_02_03.py

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

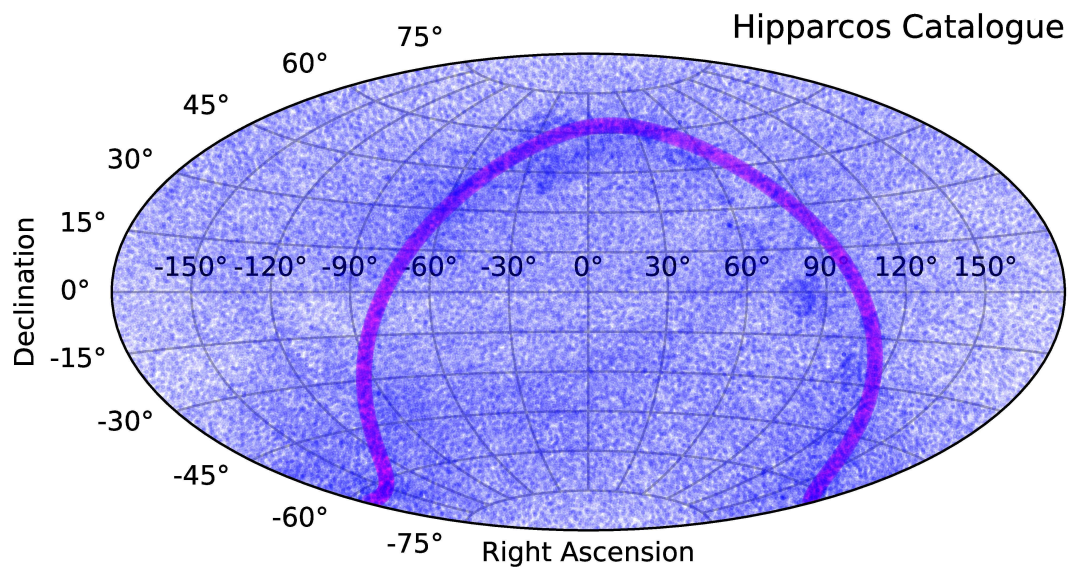


Figure 24: Positions of stars in Hipparcos Catalogue on the equatorial coordinate using Aitoff projection.

```
#
# Time-stamp: <2022/11/20 22:21:14 (CST) daisuke>
#

# importing gzip module
import gzip

# importing numpy module
import numpy

# importing astropy module
import astropy.units
import astropy.coordinates

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

# units
u_ha = astropy.units.hourangle
u_deg = astropy.units.deg
u_rad = astropy.units.rad

# files
file_cat = 'hip2.dat.gz'
file_output = 'ai202209_s10_02_03.pdf'

# list to store data
list_ra_rad = []
list_dec_rad = []

# counting number of data
n = 0

# opening file
with gzip.open(file_cat, 'rb') as fh:
```

```

# reading data
for line in fh:
    n += 1

# opening file
with gzip.open (file_cat, 'rb') as fh:
    # reading data
    i = 0
    for line in fh:
        # decoding raw byte data
        line = line.decode ('utf-8')

        # extracting data
        ra_str = line[15:28].strip ()
        dec_str = line[29:42].strip ()
        mag_str = line[129:136].strip ()
        BV_str = line[152:158].strip ()

        # skipping, if any of data is missing.
        if ( (ra_str == '') or (dec_str == '') or (mag_str == '') \
            or (BV_str == '') ):
            continue

        # conversion from string to float
        ra_rad = float (ra_str)
        dec_rad = float (dec_str)
        mag = float (mag_str)
        BV = float (BV_str)

        # coordinate
        coord = astropy.coordinates.SkyCoord (ra_rad, dec_rad, \
                                                frame=astropy.coordinates.ICRS, \
                                                unit=u_rad)

        # appending data to lists
        ra_rad_wrap = coord.ra.wrap_at (180 * u_deg).radian
        list_ra_rad.append (ra_rad_wrap)
        list_dec_rad.append (coord.dec.radian)

        # progress
        i += 1
        if (i % 5000 == 0):
            print ("progress: %6d / %6d" % (i, n) )

# making numpy arrays
data_ra_rad = numpy.array (list_ra_rad)
list_ra_rad.clear ()
data_dec_rad = numpy.array (list_dec_rad)
list_dec_rad.clear ()

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

matplotlib.rcParams['agg.path.chunksize'] = 10000

# axes
ax.grid ()
ax.set_title ('Hipparcos Catalogue', loc='right')

```

```

ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# plotting data
density = ax.hexbin (data_ra_rad, data_dec_rad, gridsize=120, \
                    cmap=matplotlib.cm.inferno)
fig.colorbar (density, ax=ax, spacing='uniform', extend='max')

# saving file
fig.savefig (file_output, bbox_inches="tight", dpi=225)

```

Execute above script to make a density plot of stars in Hipparcos Catalogue.

```

% chmod a+x ai202209_s10_02_03.py
% ./ai202209_s10_02_03.py
progress: 5000 / 117955
progress: 10000 / 117955
progress: 15000 / 117955
progress: 20000 / 117955
progress: 25000 / 117955
progress: 30000 / 117955
progress: 35000 / 117955
progress: 40000 / 117955
progress: 45000 / 117955
progress: 50000 / 117955
progress: 55000 / 117955
progress: 60000 / 117955
progress: 65000 / 117955
progress: 70000 / 117955
progress: 75000 / 117955
progress: 80000 / 117955
progress: 85000 / 117955
progress: 90000 / 117955
progress: 95000 / 117955
progress: 100000 / 117955
progress: 105000 / 117955
progress: 110000 / 117955
progress: 115000 / 117955

```

Display the PNG file. (Fig. 25)

```
% feh -dF ai202209_s10_02_03.png
```

Try following practice.

Practice 10-24

Make a Python script to make a density plot of stars in Hipparcos Catalogue on the galactic coordinate using Hammer projection.

5 Distribution of galaxies

5.1 Downloading 2MASS Extended Source Catalogue

Download the first file of 2MASS Extended Source Catalogue.

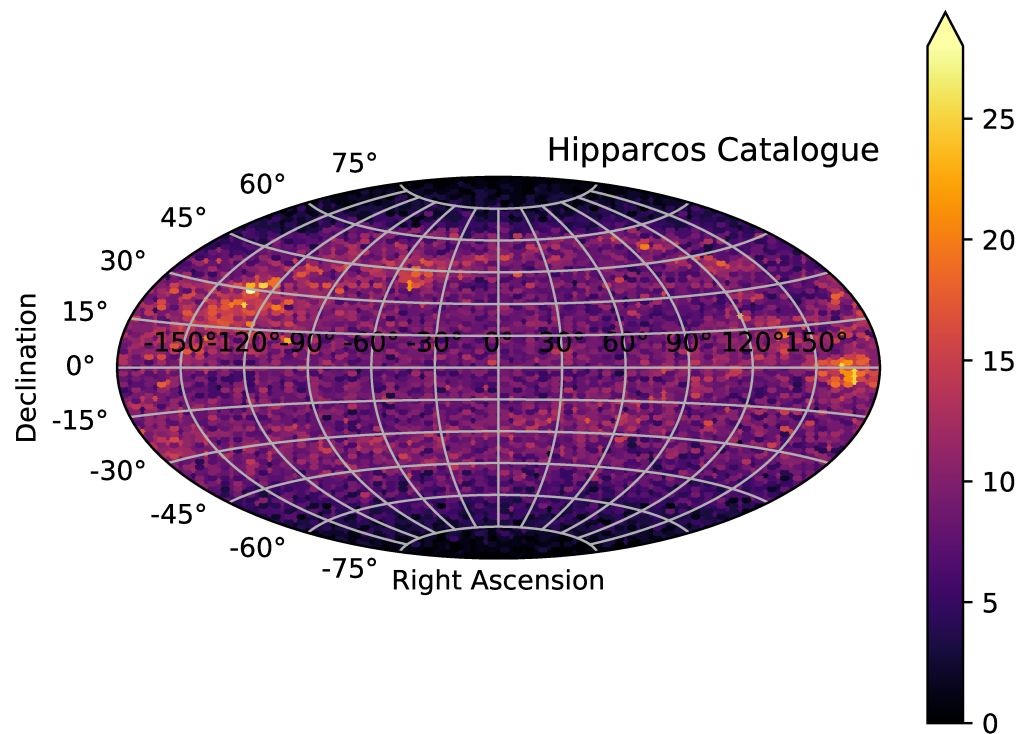


Figure 25: A density plot of stars in Hipparcos Catalogue on the equatorial coordinate using Aitoff projection.

Python Code 31: ai202209_s10_03_00.py

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

#
# Time-stamp: <2022/11/20 22:30:34 (CST) daisuke>
#

# importing urllib module
import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# URL of data file
url_data = 'https://irsa.ipac.caltech.edu/2MASS/download/allsky/xsc_aaa.gz'

# output file name
file_output = 'xsc_aaa.gz'

# printing status
print (f'Now, fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()
```

```

# printing status
print (f'Finished fetching {url_data}!')

# printing status
print (f'Now, writing the data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'wb') as fh_write:
    # writing data
    fh_write.write (data_byte)

# printing status
print (f'Finished writing the data into file "{file_output}"!')

```

Execute above script to download the first file of 2MASS Extended Source Catalogue.

```

% chmod a+x ai202209_s10_03_00.py
% ./ai202209_s10_03_00.py
Now, fetching https://irsa.ipac.caltech.edu/2MASS/download/allsky/xsc_aaa.gz...
Finished fetching https://irsa.ipac.caltech.edu/2MASS/download/allsky/xsc_aaa.gz!
Now, writing the data into file "xsc_aaa.gz"...
Finished writing the data into file "xsc_aaa.gz"!
% ls -lF xsc_aaa.gz
-rw-r--r--  1 daisuke  wheel  369484277 Nov 20 22:38 xsc_aaa.gz

```

Download the second file of 2MASS Extended Source Catalogue.

Python Code 32: ai202209_s10_03_01.py

```

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

#
# Time-stamp: <2022/11/20 22:34:08 (CST) daisuke>
#

# importing urllib module
import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# URL of data file
url_data = 'https://irsa.ipac.caltech.edu/2MASS/download/allsky/xsc_baa.gz'

# output file name
file_output = 'xsc_baa.gz'

# printing status
print (f'Now, fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()

# printing status

```

```

print (f'Finished fetching {url_data}!')

# printing status
print (f'Now, writing the data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'wb') as fh_write:
    # writing data
    fh_write.write (data_byte)

# printing status
print (f'Finished writing the data into file "{file_output}"!')

```

Execute above script to download the second file of 2MASS Extended Source Catalogue.

```

% chmod a+x ai202209_s10_03_01.py
% ./ai202209_s10_03_01.py
Now, fetching https://irsa.ipac.caltech.edu/2MASS/download/allsky/xsc_baa.gz...
Finished fetching https://irsa.ipac.caltech.edu/2MASS/download/allsky/xsc_baa.gz!
Now, writing the data into file "xsc_baa.gz"...
Finished writing the data into file "xsc_baa.gz"!
% ls -lF xsc_*
-rw-r--r--  1 daisuke  wheel   369484277 Nov 20 22:38 xsc_aaa.gz
-rw-r--r--  1 daisuke  wheel   443941833 Nov 20 22:45 xsc_baa.gz

```

Download the description of the format of 2MASS Extended Source Catalogue.

Python Code 33: ai202209_s10_03_02.py

```

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

#
# Time-stamp: <2022/11/20 22:49:15 (CST) daisuke>
#

# importing urllib module
import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# URL of data file
url_data = 'https://irsa.ipac.caltech.edu/2MASS/download/allsky/format_xsc.html'

# output file name
file_output = 'format_xsc.html'

# printing status
print (f'Now, fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()

# printing status

```

```

print (f'Finished fetching {url_data}!')

# printing status
print (f'Now, writing the data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'wb') as fh_write:
    # writing data
    fh_write.write (data_byte)

# printing status
print (f'Finished writing the data into file "{file_output}"!')

```

Execute above script to download the format description file of 2MASS Extended Source Catalogue.

```

% chmod a+x ai202209_s10_03_02.py
% ./ai202209_s10_03_02.py
Now, fetching https://irsa.ipac.caltech.edu/2MASS/download/allsky/format_xsc.html...
Finished fetching https://irsa.ipac.caltech.edu/2MASS/download/allsky/format_xsc.html...!
Now, writing the data into file "format_xsc.html"...
Finished writing the data into file "format_xsc.html"!
% ls -lF xsc_*
ls -lF xsc_* format_xsc.html
-rw-r--r--  1 daisuke  wheel      156185 Nov 20 22:49 format_xsc.html
-rw-r--r--  1 daisuke  wheel  369484277 Nov 20 22:38 xsc_aaa.gz
-rw-r--r--  1 daisuke  wheel  443941833 Nov 20 22:45 xsc_baa.gz

```

5.2 Making a density map of galaxies

Make a Python script to make a density plot of galaxies using 2MASS Extended Source Catalogue. Here is an example.

Python Code 34: ai202209_s10_03_03.py

```

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

#
# Time-stamp: <2022/11/20 23:04:43 (CST) daisuke>
#

# importing gzip module
import gzip

# importing numpy module
import numpy

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

# files
list_file_cat = ['xsc_aaa.gz', 'xsc_baa.gz']
file_output   = 'ai202209_s10_03_03.png'

# lists for storing data
list_glon_rad = []
list_glat_rad = []

# reading catalogue files
for file_cat in list_file_cat:

```

```
# printing status
print ("reading the file \"%s\"..." % (file_cat) )

# opening file
with gzip.open (file_cat, 'rb') as fh:
    # reading lines
    for line in fh:
        # decoding byte data
        line = line.decode ('utf-8')
        # splitting line
        records = line.split ('|')
        # extracting data
        glon_str = records[6].strip ()
        glat_str = records[7].strip ()
        # skip, if any of data is missing.
        if ( (glon_str == '') or (glat_str == '') ):
            continue
        # conversion from string to float
        glon_deg = float (glon_str)
        glat_deg = float (glat_str)
        # deg --> radian
        glon_rad = glon_deg / 180.0 * numpy.pi
        if (glon_rad > numpy.pi):
            glon_rad -= 2.0 * numpy.pi
        glat_rad = glat_deg / 180.0 * numpy.pi
        # appending data to lists
        list_glon_rad.append (glon_rad)
        list_glat_rad.append (glat_rad)

# printing status
print ("done!")

# printing status
print ("constructing numpy arrays...")

# making numpy arrays
data_glon_rad = numpy.array (list_glon_rad)
list_glon_rad.clear ()
data_glat_rad = numpy.array (list_glat_rad)
list_glat_rad.clear ()

# printing status
print ("done!")

# printing status
print ("generating a plot...")

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

# axes
ax.grid ()
ax.set_title ('2MASS XSC', loc='right')
ax.set_xlabel ('Galactic longitude')
ax.set_ylabel ('Galactic latitude')

# plotting data
```

```

dist = ax.hexbin (data_glon_rad, data_glat_rad, gridsize=180, \
                  cmap=matplotlib.cm.plasma)
fig.colorbar (dist, ax=ax, spacing='uniform', extend='max')

# saving file
fig.savefig (file_output, bbox_inches="tight", dpi=225)

# printing status
print ("done!")

```

Execute above script to make a density plot of galaxies using 2MASS Extended Source Catalogue.

```

% chmod a+x ai202209_s10_03_03.py
% ./ai202209_s10_03_03.py
reading the file "xsc_aaa.gz"...
done!
reading the file "xsc_baa.gz"...
done!
constructing numpy arrays...
done!
generating a plot...
done!

```

Display the PNG file. (Fig. 26)

```
% feh -dF ai202209_s10_03_03.png
```

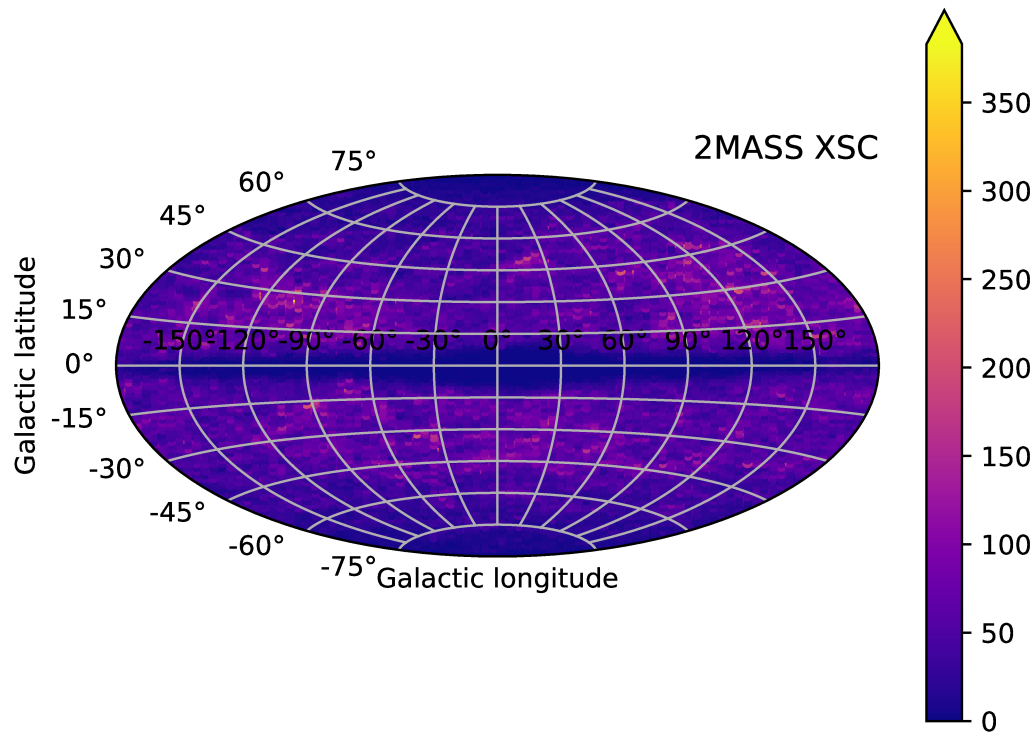


Figure 26: A density plot of galaxies in 2MASS Extended Source Catalogue on the galactic coordinate using Aitoff projection.

Try following practice.

Practice 10-25

Make a Python script to make a density plot of galaxies using 2MASS Extended Source Catalogue using Hammer projection.

6 For your further reading

Read following document to learn more about Gzip module, Astroquery, Astropy, and Matplotlib.

- Gzip module: <https://docs.python.org/3/library/gzip.html>
- Astroquery: <https://astroquery.readthedocs.io/>
 - JPL Horizons Queries: <https://astroquery.readthedocs.io/en/latest/jplhorizons/jplhorizons.html>
- Astropy: <https://www.astropy.org/>
 - Coordinates: <https://docs.astropy.org/en/stable/coordinates/>
 - SkyCoord: <https://docs.astropy.org/en/stable/api/astropy.coordinates.SkyCoord.html>
 - Transforming between Systems: <https://docs.astropy.org/en/stable/coordinates/transforming.html>
- Matplotlib: <https://matplotlib.org/stable/>
 - Projections: https://matplotlib.org/stable/api/projections_api.html
 - Geographic projections: https://matplotlib.org/stable/gallery/subplots_axes_and_figures/geo_demo.html
 - hexbin: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hexbin.html

7 Assignment

1. Histogram of V-band magnitude of stars in Hipparcos Catalogue
 - (a) Construct a histogram of V-band magnitude of stars in Hipparcos Catalogue.
 - (b) Show all the Python codes you made.
2. Distribution of star clusters
 - (a) Download open cluster catalogue from following website.
<http://cdsarc.u-strasbg.fr/ftp/cats/B/ocl/>
 - (b) Make an all-sky map of open clusters.
 - (c) Download globular cluster catalogue from following website.
http://cdsarc.u-strasbg.fr/ftp/cats/J_A+A/558/A53/
 - (d) Make an all-sky map of globular clusters.
 - (e) What you can learn from distributions of open clusters and globular clusters?
 - (f) Show all the Python codes you made.
3. Distribution of supernovae
 - (a) Download Open Supernova Catalog.
<https://github.com/astrocatalogs/supernovae>
 - (b) Show the distribution of supernovae on equatorial coordinate.
 - (c) Do you find any concentration in supernovae distribution on the sky? Why?
 - (d) Show all the Python codes you made.
4. Distribution of quasars
 - (a) Download quasar catalogue from following website.
<http://quasars.org/milliquas.htm>
 - (b) Construct an all-sky map of distribution of quasars.

- (c) Do you find any concentration in quasar distribution on the sky? Why?
- (d) Show all the Python codes you made.

5. Distribution of stars hosting exoplanets

- (a) Download exoplanet catalogue from NASA Exoplanet Archive.
<https://exoplanetarchive.ipac.caltech.edu/>
- (b) Construct an all-sky map of distribution of stars hosting exoplanets.
- (c) Is the distribution of stars hosting exoplanets uniform? Why or why not?
- (d) Show all the Python codes you made.

6. Distribution of supernova remnants

- (a) Download supernova remnant catalogue.
<https://www.mrao.cam.ac.uk/surveys/snrs/>
- (b) Construct an all-sky map of distribution of supernova remnants.
- (c) Is the distribution of supernova remnant uniform? Why or why not?
- (d) Show all the Python codes you made.