Astroinformatics 2022
Session 15: Hubble diagram and expansion of the Universe

Kinoshita Daisuke

26 December 2022
publicly accessible version

For this session, we download galaxies / supernovae catalogues, construct Hubble diagrams, and discuss the expansion of the Universe.

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

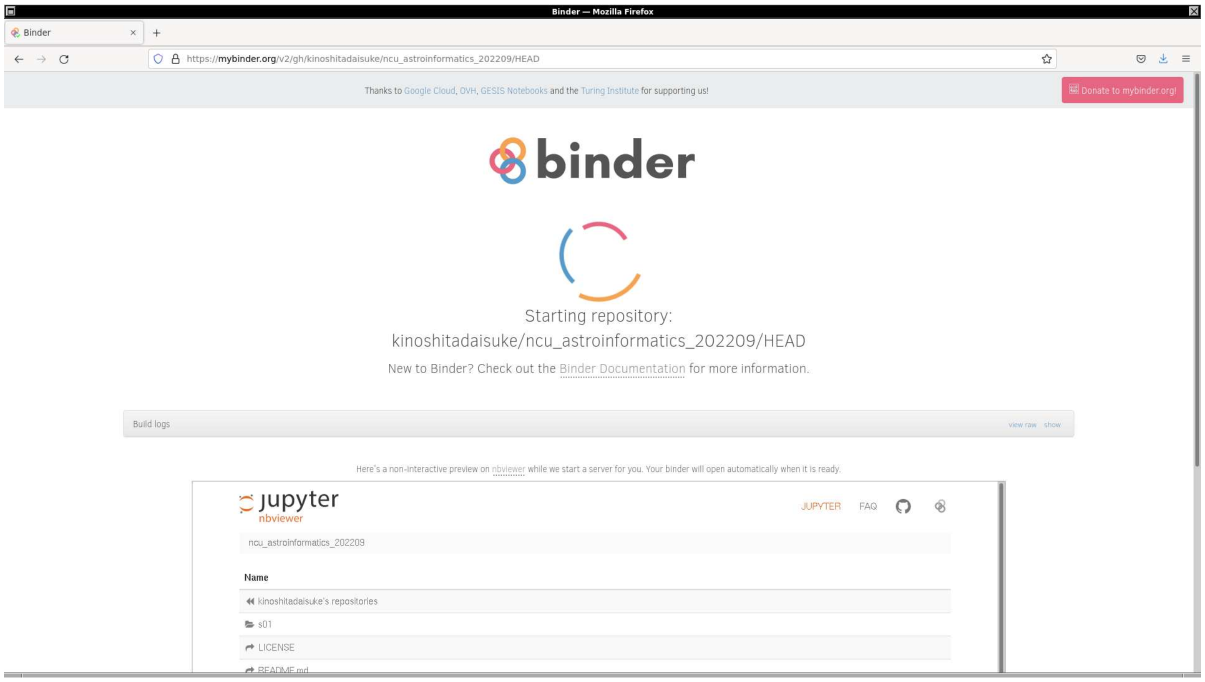
If you prefer to execute sample Python scripts for this session on JupyterLab (or Jupyter Notebook), download `.ipynb` file from GitHub repository.

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) "`s15`". (Fig. 3) Choose the file "`ai202209_s15.ipynb`" (Fig. 4 and 5) and open it (Fig. 6).



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

Thanks to Google Cloud, OVH, GESIS Notebooks and the Turing Institute for supporting us!

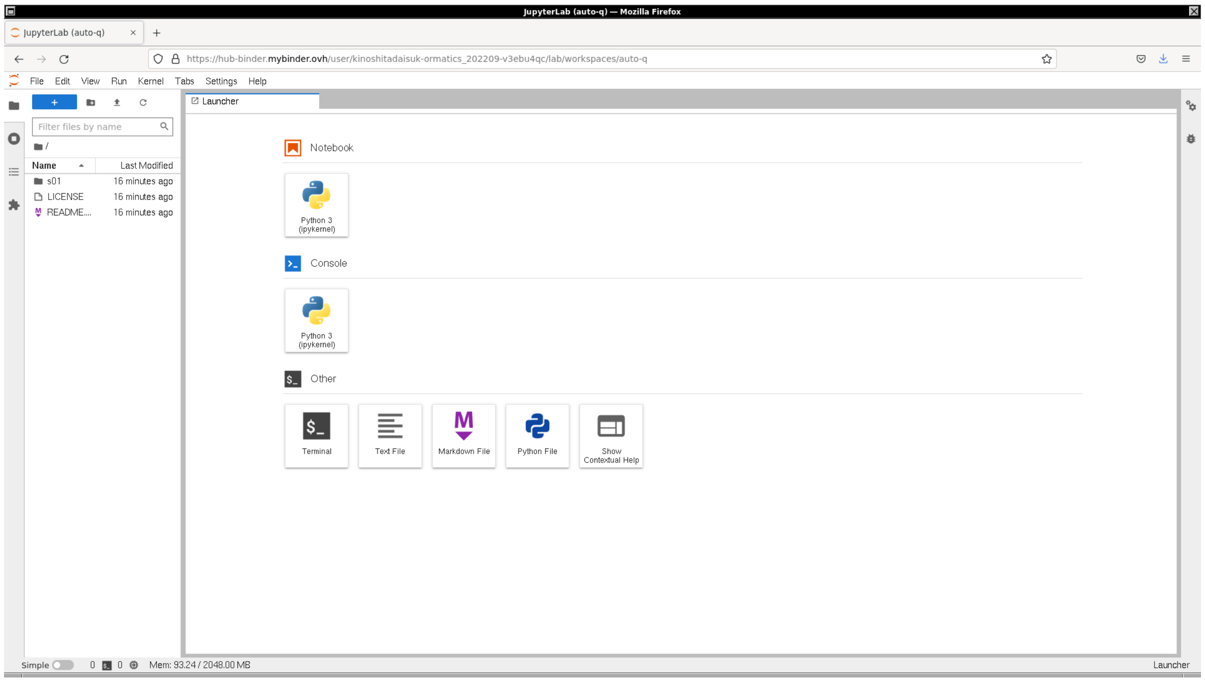
Starting repository:
kinoshitadaisuke/ncu_astroinformatics_202209/HEAD

New to Binder? Check out the Binder Documentation for more information.

Here's a non-interactive preview on nbviewer while we start a server for you. Your binder will open automatically when it is ready.

ncu_astroinformatics_202209

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

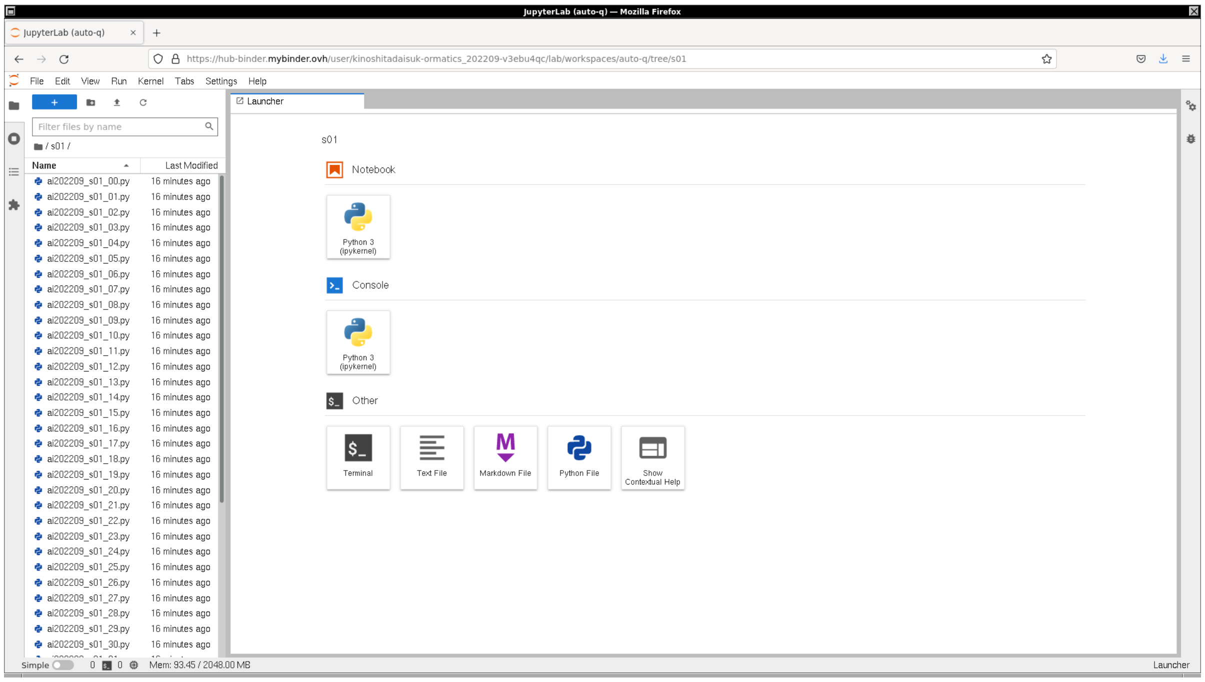


https://hub-binder.mybinder.ovh/user/kinoshitadaisuk-ormatics_202209-v3ebu4qc/lab/workspaces/auto-q

Name	Last Modified
s01	16 minutes ago
LICENSE	16 minutes ago
README....	16 minutes ago

Mem: 93.24 / 2048.00 MB

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



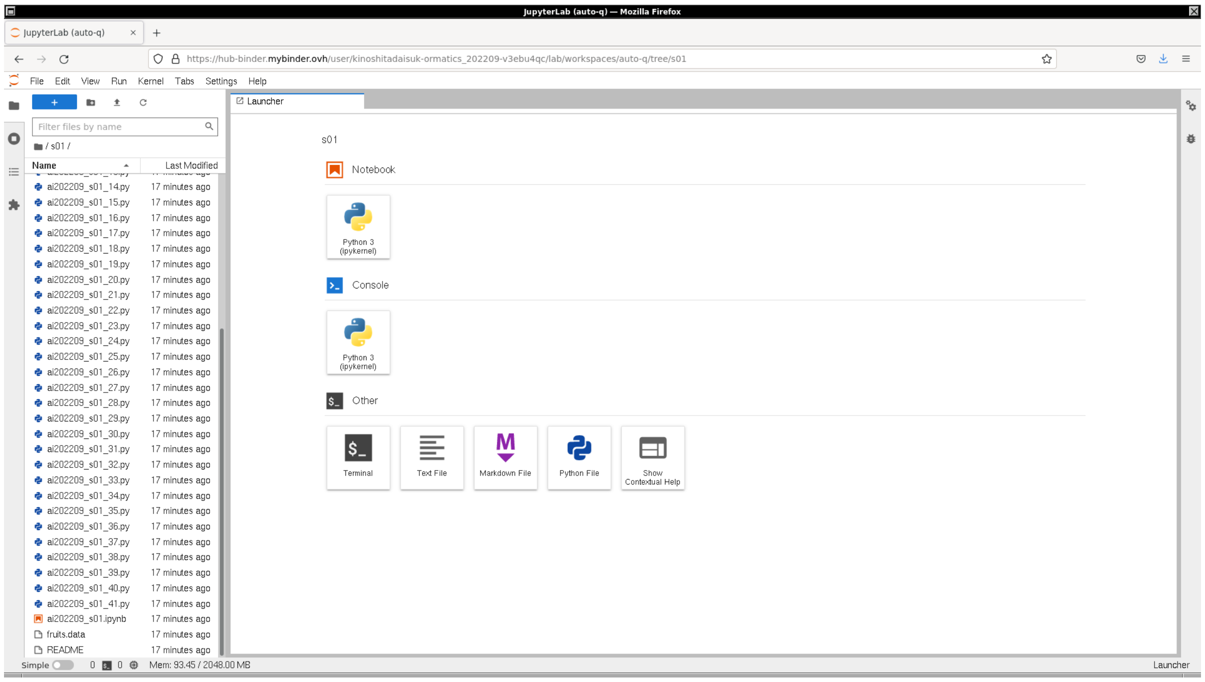
https://hub-binder.mybinder.ovh/user/kinoshitadaisuk-ormatics_202209-v3ebu4qc/lab/workspaces/auto-q/tree/s01

/ s01 /

Name	Last Modified
ai202209_s01_00.py	16 minutes ago
ai202209_s01_01.py	16 minutes ago
ai202209_s01_02.py	16 minutes ago
ai202209_s01_03.py	16 minutes ago
ai202209_s01_04.py	16 minutes ago
ai202209_s01_05.py	16 minutes ago
ai202209_s01_06.py	16 minutes ago
ai202209_s01_07.py	16 minutes ago
ai202209_s01_08.py	16 minutes ago
ai202209_s01_09.py	16 minutes ago
ai202209_s01_10.py	16 minutes ago
ai202209_s01_11.py	16 minutes ago
ai202209_s01_12.py	16 minutes ago
ai202209_s01_13.py	16 minutes ago
ai202209_s01_14.py	16 minutes ago
ai202209_s01_15.py	16 minutes ago
ai202209_s01_16.py	16 minutes ago
ai202209_s01_17.py	16 minutes ago
ai202209_s01_18.py	16 minutes ago
ai202209_s01_19.py	16 minutes ago
ai202209_s01_20.py	16 minutes ago
ai202209_s01_21.py	16 minutes ago
ai202209_s01_22.py	16 minutes ago
ai202209_s01_23.py	16 minutes ago
ai202209_s01_24.py	16 minutes ago
ai202209_s01_25.py	16 minutes ago
ai202209_s01_26.py	16 minutes ago
ai202209_s01_27.py	16 minutes ago
ai202209_s01_28.py	16 minutes ago
ai202209_s01_29.py	16 minutes ago
ai202209_s01_30.py	16 minutes ago

0 0 Mem: 93.45 / 2048.00 MB

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



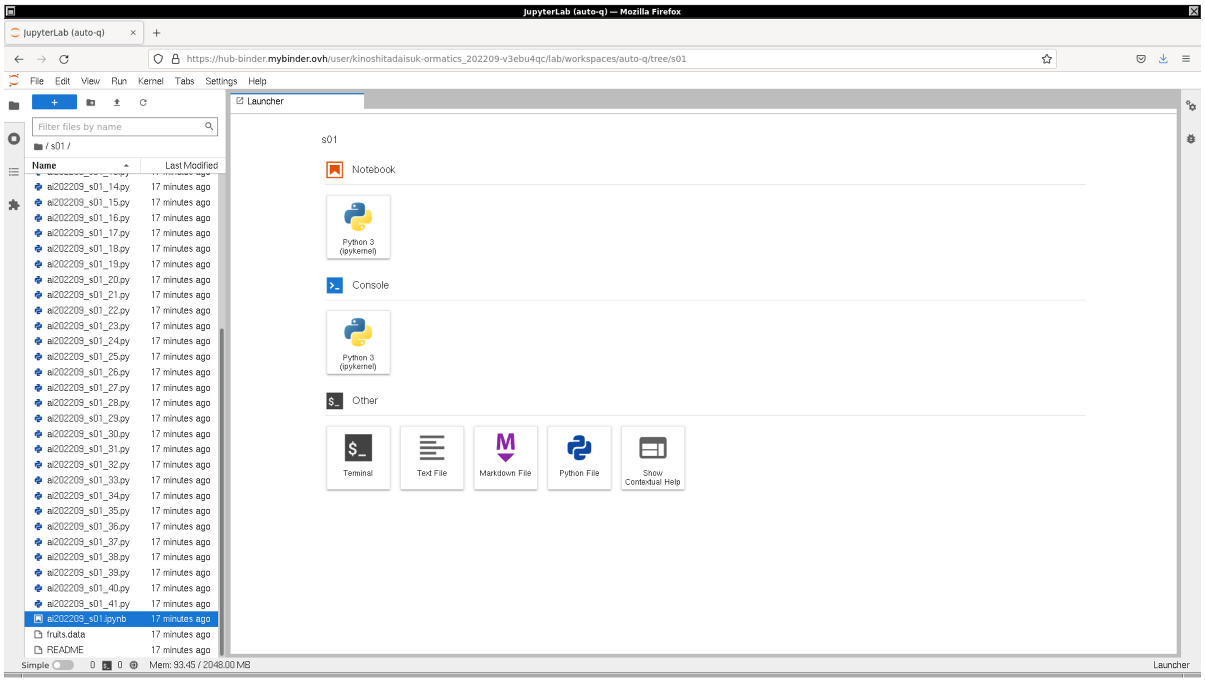
https://hub-binder.mybinder.ovh/user/kinoshitadaisuk-ormatics_202209-v3ebu4qc/lab/workspaces/auto-q/tree/s01

/ s01 /

Name	Last Modified
ai202209_s01_14.py	17 minutes ago
ai202209_s01_15.py	17 minutes ago
ai202209_s01_16.py	17 minutes ago
ai202209_s01_17.py	17 minutes ago
ai202209_s01_18.py	17 minutes ago
ai202209_s01_19.py	17 minutes ago
ai202209_s01_20.py	17 minutes ago
ai202209_s01_21.py	17 minutes ago
ai202209_s01_22.py	17 minutes ago
ai202209_s01_23.py	17 minutes ago
ai202209_s01_24.py	17 minutes ago
ai202209_s01_25.py	17 minutes ago
ai202209_s01_26.py	17 minutes ago
ai202209_s01_27.py	17 minutes ago
ai202209_s01_28.py	17 minutes ago
ai202209_s01_29.py	17 minutes ago
ai202209_s01_30.py	17 minutes ago
ai202209_s01_31.py	17 minutes ago
ai202209_s01_32.py	17 minutes ago
ai202209_s01_33.py	17 minutes ago
ai202209_s01_34.py	17 minutes ago
ai202209_s01_35.py	17 minutes ago
ai202209_s01_36.py	17 minutes ago
ai202209_s01_37.py	17 minutes ago
ai202209_s01_38.py	17 minutes ago
ai202209_s01_39.py	17 minutes ago
ai202209_s01_40.py	17 minutes ago
ai202209_s01_41.py	17 minutes ago
ai202209_s01.ipynb	17 minutes ago
fruits.data	17 minutes ago
README	17 minutes ago

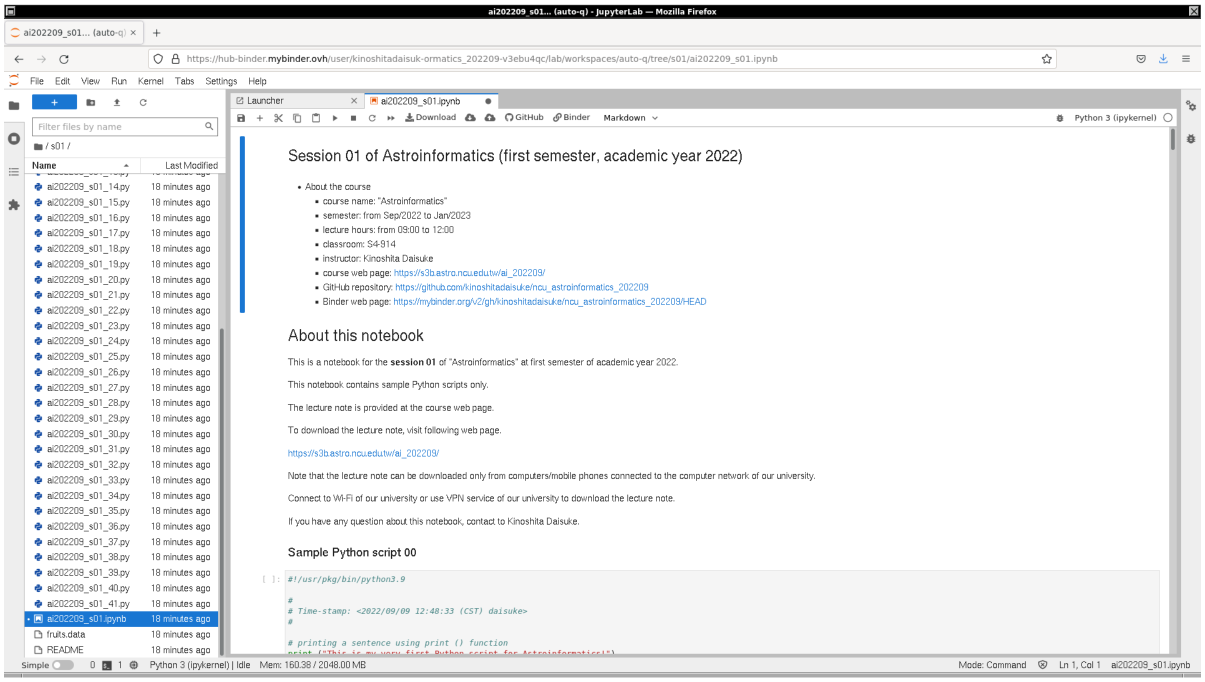
0 0 Mem: 93.45 / 2048.00 MB

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


JupyterLab (auto-q) — Mozilla Firefox
https://hub-binder.mybinder.ovh/user/kinoshitadaisuk-ormatics_202209-v3ebu4qc/lab/workspaces/auto-q/tree/s01
File Edit View Run Kernel Tabs Settings Help
Launcher
s01
Notebook
Python 3 (ipykernel)
Console
Python 3 (ipykernel)
Other
Terminal
Text File
Markdown File
Python File
Show Contextual Help
Mem: 93.45 / 2048.00 MB

Name	Last Modified
ai202209_s01_14.py	17 minutes ago
ai202209_s01_15.py	17 minutes ago
ai202209_s01_16.py	17 minutes ago
ai202209_s01_17.py	17 minutes ago
ai202209_s01_18.py	17 minutes ago
ai202209_s01_19.py	17 minutes ago
ai202209_s01_20.py	17 minutes ago
ai202209_s01_21.py	17 minutes ago
ai202209_s01_22.py	17 minutes ago
ai202209_s01_23.py	17 minutes ago
ai202209_s01_24.py	17 minutes ago
ai202209_s01_25.py	17 minutes ago
ai202209_s01_26.py	17 minutes ago
ai202209_s01_27.py	17 minutes ago
ai202209_s01_28.py	17 minutes ago
ai202209_s01_29.py	17 minutes ago
ai202209_s01_30.py	17 minutes ago
ai202209_s01_31.py	17 minutes ago
ai202209_s01_32.py	17 minutes ago
ai202209_s01_33.py	17 minutes ago
ai202209_s01_34.py	17 minutes ago
ai202209_s01_35.py	17 minutes ago
ai202209_s01_36.py	17 minutes ago
ai202209_s01_37.py	17 minutes ago
ai202209_s01_38.py	17 minutes ago
ai202209_s01_39.py	17 minutes ago
ai202209_s01_40.py	17 minutes ago
ai202209_s01_41.py	17 minutes ago
ai202209_s01.ipynb	17 minutes ago
fruits.data	17 minutes ago
README	17 minutes ago

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


ai202209_s01... (auto-q) - JupyterLab — Mozilla Firefox
https://hub-binder.mybinder.ovh/user/kinoshitadaisuk-ormatics_202209-v3ebu4qc/lab/workspaces/auto-q/tree/s01/ai202209_s01.ipynb
File Edit View Run Kernel Tabs Settings Help
Launcher
ai202209_s01.ipynb
Download GitHub Binder Markdown
Python 3 (ipykernel)
Python 3 (ipykernel) | Idle Mem: 160.38 / 2048.00 MB
Mode: Command Ln 1, Col 1 ai202209_s01.ipynb

Name	Last Modified
ai202209_s01_14.py	18 minutes ago
ai202209_s01_15.py	18 minutes ago
ai202209_s01_16.py	18 minutes ago
ai202209_s01_17.py	18 minutes ago
ai202209_s01_18.py	18 minutes ago
ai202209_s01_19.py	18 minutes ago
ai202209_s01_20.py	18 minutes ago
ai202209_s01_21.py	18 minutes ago
ai202209_s01_22.py	18 minutes ago
ai202209_s01_23.py	18 minutes ago
ai202209_s01_24.py	18 minutes ago
ai202209_s01_25.py	18 minutes ago
ai202209_s01_26.py	18 minutes ago
ai202209_s01_27.py	18 minutes ago
ai202209_s01_28.py	18 minutes ago
ai202209_s01_29.py	18 minutes ago
ai202209_s01_30.py	18 minutes ago
ai202209_s01_31.py	18 minutes ago
ai202209_s01_32.py	18 minutes ago
ai202209_s01_33.py	18 minutes ago
ai202209_s01_34.py	18 minutes ago
ai202209_s01_35.py	18 minutes ago
ai202209_s01_36.py	18 minutes ago
ai202209_s01_37.py	18 minutes ago
ai202209_s01_38.py	18 minutes ago
ai202209_s01_39.py	18 minutes ago
ai202209_s01_40.py	18 minutes ago
ai202209_s01_41.py	18 minutes ago
ai202209_s01.ipynb	18 minutes ago
fruits.data	18 minutes ago
README	18 minutes ago

Session 01 of Astroinformatics (first semester, academic year 2022)

- About the course
  - course name: "Astroinformatics"
  - semester: from Sep/2022 to Jan/2023
  - lecture hours: from 09:00 to 12:00
  - classroom: S4-914
  - instructor: Kinoshita Daisuke
  - course web page: https://s3b.astro.ncu.edu.tw/ai_202209/
  - GitHub repository: https://github.com/kinoshitadaisuke/ncu_astroinformatics_202209
  - Binder web page: https://mybinder.org/v2/gh/kinoshitadaisuke/ncu_astroinformatics_202209/HEAD

About this notebook

This is a notebook for the **session 01** of "Astroinformatics" at first semester of academic year 2022.

This notebook contains sample Python scripts only.

The lecture note is provided at the course web page.

To download the lecture note, visit following web page.

https://s3b.astro.ncu.edu.tw/ai_202209/

Note that the lecture note can be downloaded only from computers/mobile phones connected to the computer network of our university.

Connect to Wi-Fi of our university or use VPN service of our university to download the lecture note.

If you have any question about this notebook, contact to Kinoshita Daisuke.

Sample Python script 00

```
[ ]: #!/usr/pkg/bin/python3.9

     #
     # Time-stamp: <2022/09/09 12:48:33 (CST) daisuke>
     #

     # printing a sentence using print () function
```

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

2 Check of availability of required Python packages

2.1 Availability check of Astropy package

For this session, "`astropy`" package is needed. Check whether you have "`astropy`" package properly installed on your computer.

Python Code 1: ai202209_s15_00_00.py

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

#
# Time-stamp: <2022/12/25 11:38:27 (CST) daisuke>
#

# check of availability of astropy module
try:
    # importing astropy module
    import astropy
except:
    # if astropy module is not installed, print an error message
    print (f"The module 'astropy' is not installed on your computer.")
    print (f"The module 'astropy' is required for this session.")
    print (f"Visit following web page and install the package 'astropy'.")
    print (f"  https://docs.astropy.org/")
    print (f"After the installation, try to run this script again.")
else:
    # if astropy module is found, print following message
    print (f"The module 'astropy' is found on your computer.")
finally:
    # print that the check of availability of astropy module is finished
    print (f"An availability check of 'astropy' module is now finished.")
```

Execute above script to check the availability of "`astropy`" package. If you do not have "`astropy`" package installed on your computer, you see following message.

```
% chmod a+x ai202209_s15_00_00.py
% ./ai202209_s15_00_00.py
The module 'astropy' is not installed on your computer.
The module 'astropy' is required for this session.
Visit following web page and install the package 'astropy'.
  https://docs.astropy.org/
After the installation, try to run this script again.
An availability check of 'astropy' module is now finished.
```

If you see above message, install "`astropy`" package, and try to execute the script again after the installation.
If you have "`astropy`" package, you see following message.

```
% ./ai202209_s15_00_00.py
The module 'astropy' is found on your computer.
An availability check of 'astropy' module is now finished.
```

About the exception handling, read the Section "Exceptions" of the session 01 "Basic Python Programming".
Try following practice.

Practice 15-01

Make a Python script to check the availability of "`matplotlib`" module.

2.2 More flexible Python script for availability check of packages

Make a more flexible Python script for availability check of Python packages. Use "`argparse`" module. Here is an example.

Python Code 2: ai202209_s15_00_01.py

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

#
# Time-stamp: <2022/12/25 11:42:59 (CST) daisuke>
#

# importing argparse module
import argparse

#
# command-line argument analysis
#

# constructing parser object
desc = f"availability check of Python modules"
parser = argparse.ArgumentParser (description=desc)

# adding options
parser.add_argument ('module', type=str, nargs='+', \
                     help=f"module name (e.g. numpy)")

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

#
# input parameters
#

# list of module names for availability check
list_modules = args.module

#
# availability check of modules
#

for module in list_modules:
    # check of availability of rebound module
    try:
        # importing module
        imported = __import__ (module)
    except:
        # if rebound module is not installed, print an error message
        print (f"The module '{module}' is NOT installed on your computer.")
    else:
        # if rebound module is found, print following message
        print (f"The module '{module}' is found on your computer.")
        print (f"{imported}")
    finally:
        # print that the check of availability of rebound module is finished
        print (f"An availability check of '{module}' module is now finished.")
        print (f"")
```

Execute above script to check availability of Astropy package.

```
% chmod a+x ai202209_s15_00_01.py
% ./ai202209_s15_00_01.py -h
usage: ai202209_s15_00_01.py [-h] module [module ...]

availability check of Python modules

positional arguments:
  module      module name (e.g. numpy)

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

% ./ai202209_s15_00_01.py astropy
The module 'astropy' is found on your computer.
<module 'astropy' from '/usr/pkg/lib/python3.9/site-packages/astropy/__init__.py'>
An availability check of 'astropy' module is now finished.
```

Execute the script to check availability of both "`pint`" module, "`uncertainties`" module, and "`sklearn`" module.

```
% ./ai202209_s15_00_01.py pint uncertainties sklearn
./ai202209_s15_00_01.py pint uncertainties sklearn
The module 'pint' is NOT installed on your computer.
An availability check of 'pint' module is now finished.

The module 'uncertainties' is NOT installed on your computer.
An availability check of 'uncertainties' module is now finished.

The module 'sklearn' is NOT installed on your computer.
An availability check of 'sklearn' module is now finished.
```

If you have these packages installed on your computer, then you see following messages.

```
% ./ai202209_s15_00_01.py pint uncertainties sklearn
The module 'pint' is found on your computer.
<module 'pint' from '/usr/lib/python3/dist-packages/pint/__init__.py'>
An availability check of 'pint' module is now finished.

The module 'uncertainties' is found on your computer.
<module 'uncertainties' from '/usr/lib/python3/dist-packages/uncertainties/__init
__.py'>
An availability check of 'uncertainties' module is now finished.

The module 'sklearn' is NOT installed on your computer.
An availability check of 'sklearn' module is now finished.
```

Try following practice.

Practice 15-02

Make your own Python script to check the availability of Python modules. Use "`argparse`" module.

3 Reading a CSV file

For this session, we need to read a CSV file.

3.1 Downloading a CSV file

Visit following repository.

- HYG star database: `https://github.com/astronexus/HYG-Database`

Download the file "`hygdata_v3.csv`". To download the file using a Python script, try following.

Python Code 3: ai202209_s15_01_00.py

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

#
# Time-stamp: <2022/12/25 12:18:02 (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://raw.githubusercontent.com/astronexus/HYG-Database/' \
    + 'master/hygdata_v3.csv'

# output file name
file_output = 'hygdata_v3.csv'

# 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 a CSV file.

```
% chmod a+x ai202209_s15_01_00.py
% ./ai202209_s15_01_00.py
Now, fetching https://raw.githubusercontent.com/astronexus/HYG-Database/master/hy
gdata_v3.csv...
Finished fetching https://raw.githubusercontent.com/astronexus/HYG-Database/maste
```

```
r/hygdata_v3.csv!
Now, writing the data into file "hygdata_v3.csv"...
Finished writing the data into file "hygdata_v3.csv"!
% ls -lF *.csv
-rw-r--r--  1 daisuke  taiwan  33449663 Dec 25 12:18 hygdata_v3.csv
% head hygdata_v3.csv | cut -b 1-80
id,hip,hd,hr,gl,bf,proper,ra,dec,dist,pmra,pmdec,rv,mag,absmag,spect,ci,x,y,z,vx
0,,,,,,Sol,0.000000,0.000000,0.0000,0.00,0.00,0.0,-26.700,4.850,G2V,0.656,0.0000
1,1,224700,,,,,0.000060,1.089009,219.7802,-5.20,-1.88,0.0,9.100,2.390,F5,0.482,2
2,2,224690,,,,,0.000283,-19.498840,47.9616,181.21,-0.93,0.0,9.270,5.866,K3V,0.99
3,3,224699,,,,,0.000335,38.859279,442.4779,5.24,-2.91,0.0,6.610,-1.619,B9,-0.019
4,4,224707,,,,,0.000569,-51.893546,134.2282,62.85,0.16,0.0,8.060,2.421,F0V,0.370
5,5,224705,,,,,0.000665,-40.591202,257.7320,2.53,9.07,0.0,8.550,1.494,G8III,0.90
6,6,,,,,,0.001246,3.946458,55.0358,226.29,-12.84,0.0,12.310,8.607,M0V:,1.336,54.
7,7,,,,,,0.001470,20.036114,57.8704,-208.12,-200.79,0.0,9.640,5.828,G0,0.740,54.
8,8,224709,,,,,0.001823,25.886461,200.8032,19.09,-5.66,-31.0,9.050,2.536,M6e-M8.
```

Try following practice.

Practice 15-03

Make your own Python script to download a file from a remote host.

3.2 Reading a CSV file using astropy.io.ascii

Make a Python script to read a CSV file using "`astropy.io.ascii`" module. Here is an example.

Python Code 4: ai202209_s15_01_01.py

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

#
# Time-stamp: <2022/12/25 12:27:51 (CST) daisuke>
#

# importing astropy module
import astropy.io.ascii

# CSV file name
file_csv = 'hygdata_v3.csv'

# reading a CSV file and storing data in an astropy table
table = astropy.io.ascii.read (file_csv, format='csv')

# printing astropy table
print (table)
```

Execute above script to read a CSV file and store data in an Astropy table.

```
% chmod a+x ai202209_s15_01_01.py
% ./ai202209_s15_01_01.py
  id   hip   hd    hr    gl    ...          lum           var var_min var_max
------ --- ------ --- -------  ... ---------------------- --- ------- -------
     0  --     --  --      --  ...                    1.0  --      --      --
     1   1 224700  --      --  ...      9.638290236239703  --      --      --
     2   2 224690  --      --  ...    0.39228346253952057  --      --      --
     3   3 224699  --      --  ...      386.9011316551087  --      --      --
     4   4 224707  --      --  ...      9.366988779521161  --      --      --
     5   5 224705  --      --  ...     21.998851090492494  --      --      --
     6    6      --  --       -- ...     0.03141955284641073   --   12.462   12.162
   ... ...     ... ...       ... ...                     ... ...      ...      ...
119609  --      --  -- GJ 1292 ...   0.003435579478998743   --       --       --
119610  --      --  -- NN 4380 ...   0.001799699516577357   --       --       --
119611  --      --  -- NN 4381 ...   0.002199885109049247   --       --       --
119612  --      --  -- NN 4385 ...  8.838936217394057e-05   --       --       --
119613  --      --  -- NN 4386 ...   0.002224333998485111   --       --       --
119614  --      --  -- NN 4387 ...  0.0010864256236170645   --       --       --
119615  --      --  --  Gl 915 ... 0.00031944787242086883   --       --       --
Length = 119614 rows
```

To learn more about the way to read a CSV file using Astropy, visit following web page and read the document.

- Astropy: `https://www.astropy.org/`
    - "ASCII Tables": `https://docs.astropy.org/en/stable/io/ascii/`

Try following practice.

Practice 15-04

Make your own Python script to read a CSV file and store the data into an Astropy table using "`astropy.io.ascii`" module.

Print the column names of the Astropy table.

Python Code 5: ai202209_s15_01_02.py

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

#
# Time-stamp: <2022/12/25 12:53:10 (CST) daisuke>
#

# importing astropy module
import astropy.io.ascii

# CSV file name
file_csv = 'hygdata_v3.csv'

# reading a CSV file and storing data in an astropy table
table = astropy.io.ascii.read (file_csv, format='csv')

# printing column names of astropy table
print (table.info ())
```

Execute above script to read a CSV file and print the column names of the table.

```
% chmod a+x ai202209_s15_01_02.py
% ./ai202209_s15_01_02.py
<Table length=119614>
    name      dtype      class     n_bad
------------ ------- ------------ ------
          id   int64       Column      0
         hip   int64 MaskedColumn   1658
          hd   int64 MaskedColumn  20732
          hr   int64 MaskedColumn 110589
          gl    str9 MaskedColumn 115813
          bf   str10 MaskedColumn 116516
      proper   str24 MaskedColumn 119468
```

```
          ra float64       Column      0
         dec float64       Column      0
        dist float64       Column      0
        pmra float64       Column      0
       pmdec float64       Column      0
          rv float64       Column      0
         mag float64       Column      0
      absmag float64       Column      0
       spect   str12 MaskedColumn   3050
          ci float64 MaskedColumn   1882
           x float64       Column      0
           y float64       Column      0
           z float64       Column      0
          vx float64       Column      0
          vy float64       Column      0
          vz float64       Column      0
       rarad float64       Column      0
      decrad float64       Column      0
     pmrarad float64       Column      0
    pmdecrad float64       Column      0
       bayer    str5 MaskedColumn 118078
        flam   int64 MaskedColumn 116874
         con    str3 MaskedColumn   1630
        comp   int64       Column      0
comp_primary   int64       Column      0
        base    str8 MaskedColumn 118527
         lum float64       Column      0
         var    str5 MaskedColumn 113624
     var_min float64 MaskedColumn 102623
     var_max float64 MaskedColumn 102623
None
```

To learn more about the way to deal with an Astropy table, visit following web page and read the document.

- Astropy: `https://www.astropy.org/`
  - "Data Tables": `https://docs.astropy.org/en/stable/table/`

Try following practice.

Practice 15-05

Make your own Python script to print column names of an Astropy table.

Make a Python script to read the CSV file and print the information of the star Vega.

Python Code 6: ai202209_s15_01_03.py

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

#
# Time-stamp: <2022/12/25 12:51:22 (CST) daisuke>
#

# importing astropy module
import astropy.io.ascii

# CSV file name
file_csv = 'hygdata_v3.csv'

# reading a CSV file and storing data in an astropy table
```

```
table = astropy.io.ascii.read (file_csv, format='csv')

# making a mask for Vega
obj  = 'Vega'
mask = (table['proper'] == obj)

# printing information of Vega
print (f"object name = {obj}:")
print (f"{table[mask]['con', 'ra', 'dec', 'mag', 'dist', 'absmag', 'spect']}")
```

Execute above script to read a CSV file and print the information of the star Vega.

```
% chmod a+x ai202209_s15_01_03.py
% ./ai202209_s15_01_03.py
object name = Vega:
con    ra        dec     mag   dist  absmag spect
--- -------- --------- ---- ------ ------ ------
Lyr 18.61564 38.783692 0.03 7.6787  0.604 A0Vvar
```

Try following practice.

Practice 15-06

Make your own Python script to print the information of the star Sirius.

4 Reading a JSON file

For this session, we need to read a JSON file.

4.1 Downloading a JSON file

Visit following repository.

- exoplanets: `https://github.com/paulfitz/exoplanets`

Download the file "`exoplanet.json`". To download the file using a Python script, try following.

Python Code 7: ai202209_s15_02_00.py

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

#
# Time-stamp: <2022/12/25 13:21:19 (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://raw.githubusercontent.com/paulfitz/exoplanets/' \
    + 'master/data/exoplanet.json'

# output file name
```

```
file_output = 'exoplanet.json'

# 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 a JSON file.

```
% chmod a+x ai202209_s15_02_00.py
% ./ai202209_s15_02_00.py
Now, fetching https://raw.githubusercontent.com/paulfitz/exoplanets/master/data/e
xoplanet.json...
Finished fetching https://raw.githubusercontent.com/paulfitz/exoplanets/master/da
ta/exoplanet.json!
Now, writing the data into file "exoplanet.json"...
Finished writing the data into file "exoplanet.json"!
% ls -lF *.json
-rw-r--r--  1 daisuke  taiwan  11071049 Dec 25 13:23 exoplanet.json
% head -20 exoplanet.json
[
  {
    "star_alternate_names": null,
    "star_magnetic_field": null,
    "star_detected_disc": null,
    "star_teff_error_max": null,
    "star_teff_error_min": null,
    "star_teff": null,
    "star_age_error_max": null,
    "star_age_error_min": null,
    "star_age": null,
    "star_sp_type": null,
    "star_radius_error_max": null,
    "star_radius_error_min": null,
    "star_radius": null,
    "star_mass_error_max": 0.007,
    "star_mass_error_min": 0.007,
    "star_mass": 0.048,
    "star_metallicity_error_max": null,
    "star_metallicity_error_min": null,
```

Try following practice.

Practice 15-07

Make your own Python script to download a file from a remote host. Use `argparse` module to accept command-line arguments.

4.2 Reading a JSON file

Make a Python script to read a JSON file. Here is an example.

Python Code 8: ai202209_s15_02_01.py

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

#
# Time-stamp: <2022/12/25 19:02:52 (CST) daisuke>
#

# importing json module
import json

# JSON file name
file_json = 'exoplanet.json'

# opening file
with open (file_json, 'r') as fh:
    # reading JSON file
    data = json.load (fh)

# printing keys of the data
for key in data[0].keys ():
    print (f"{key}")
```

Execute above script to read a JSON file and print keys of the dictionary. The data is stored in a Python's dictionary.

```
% chmod a+x ai202209_s15_02_01.py
% ./ai202209_s15_02_01.py
star_alternate_names
star_magnetic_field
star_detected_disc
star_teff_error_max
star_teff_error_min
star_teff
star_age_error_max
star_age_error_min
star_age
star_sp_type
star_radius_error_max
star_radius_error_min
star_radius
star_mass_error_max
star_mass_error_min
star_mass
star_metallicity_error_max
star_metallicity_error_min
star_metallicity
star_distance_error_max
star_distance_error_min
star_distance
mag_k
mag_h
```

```
mag_j
mag_i
mag_v
dec
ra
star_name
molecules
alternate_names
radius_detection_type
mass_detection_type
tperi_error_max
tperi_error_min
tperi
omega_error_max
omega_error_min
omega
updated
discovered
angular_distance
inclination_error_max
inclination_error_min
inclination
eccentricity_error_max
eccentricity_error_min
eccentricity
semi_major_axis_error_max
mass_sini_error_max
mass_sini_error_min
mass_sini
mass_error_max
mass_error_min
mass
planet_status
# name
radius
radius_error_min
radius_error_max
orbital_period
orbital_period_error_min
orbital_period_error_max
semi_major_axis
semi_major_axis_error_min
tconj
tconj_error_min
tconj_error_max
tzero_tr
tzero_tr_error_min
tzero_tr_error_max
tzero_tr_sec
tzero_tr_sec_error_min
tzero_tr_sec_error_max
lambda_angle
lambda_angle_error_min
lambda_angle_error_max
impact_parameter
impact_parameter_error_min
impact_parameter_error_max
tzero_vr
tzero_vr_error_min
```

```
tzero_vr_error_max
k
k_error_min
k_error_max
temp_calculated
temp_calculated_error_min
temp_calculated_error_max
temp_measured
hot_point_lon
geometric_albedo
geometric_albedo_error_min
geometric_albedo_error_max
log_g
publication_status
detection_type
```

Try following practice.

Practice 15-08

Make your own Python script to read a JSON file.

Make a Python script to print the information of exoplanets hosted by the star 51 Peg. Here is an example.

Python Code 9: ai202209_s15_02_02.py

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

#
# Time-stamp: <2022/12/25 19:16:57 (CST) daisuke>
#

# importing json module
import json

# JSON file name
file_json = 'exoplanet.json'

# opening file
with open (file_json, 'r') as fh:
    # reading JSON file
    data = json.load (fh)

# printing the information of exoplanet hosted by 51 Peg
for i in range (len (data)):
    if (data[i]['star_name'] == '51 Peg'):
        print (f"Host star: {data[i]['star_name']}")
        print (f"  {data[i]['# name']:12s}:", \
               f" M={data[i]['mass']:5.2f},", \
               f" P={data[i]['orbital_period']:8.4f},", \
               f" a={data[i]['semi_major_axis']:8.4f},", \
               f" detect={data[i]['detection_type']}")
```

Execute above script to read a JSON file and print the information of exoplanet hosted by 51 Peg.

```
% chmod a+x ai202209_s15_02_02.py
% ./ai202209_s15_02_02.py
Host star: 51 Peg
  51 Peg b    :  M= 0.47,  P=  4.2308,  a=  0.0520,  detect=Radial Velocity
```

Try following practice.

Practice 15-09

Make your own Python script to read a JSON file and print keys of the dictionary.

Make a Python script to print the information of exoplanets hosted by the star Kepler-90. Here is an example.

Python Code 10: ai202209_s15_02_03.py

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

#
# Time-stamp: <2022/12/25 19:23:36 (CST) daisuke>
#

# importing json module
import json

# JSON file name
file_json = 'exoplanet.json'

# opening file
with open (file_json, 'r') as fh:
    # reading JSON file
    data = json.load (fh)

# printing the information of exoplanet hosted by Kepler-90
for i in range (len (data)):
    if (data[i]['star_name'] == 'Kepler-90'):
        print (f"Host star: {data[i]['star_name']}")
        print (f"  {data[i]['# name']:12s}:", \
               f" M={data[i]['mass']},", \
               f" P={data[i]['orbital_period']:8.4f},", \
               f" a={data[i]['semi_major_axis']:8.4f},", \
               f" detect={data[i]['detection_type']}")
```

Execute above script to read a JSON file and print the information of exoplanet hosted by Kepler-90.

```
% chmod a+x ai202209_s15_02_03.py
% ./ai202209_s15_02_03.py
Host star: Kepler-90
  Kepler-90 b :  M=None,  P=  7.0082,  a=  0.0740,  detect=Primary Transit
Host star: Kepler-90
  Kepler-90 c :  M=None,  P=  8.7194,  a=  0.0890,  detect=Primary Transit
Host star: Kepler-90
  Kepler-90 d :  M=None,  P= 59.7367,  a=  0.3200,  detect=Primary Transit
Host star: Kepler-90
  Kepler-90 e :  M=None,  P= 91.9391,  a=  0.4200,  detect=Primary Transit
Host star: Kepler-90
  Kepler-90 f :  M=None,  P=124.9144,  a=  0.4800,  detect=Primary Transit
Host star: Kepler-90
  Kepler-90 g :  M=None,  P=210.6070,  a=  0.7100,  detect=Primary Transit
Host star: Kepler-90
  Kepler-90 h :  M=None,  P=331.6006,  a=  1.0100,  detect=Primary Transit
Host star: Kepler-90
  Kepler-90 i :  M=None,  P= 14.4491,  a=  0.2000,  detect=Primary Transit
```

Try following practice.

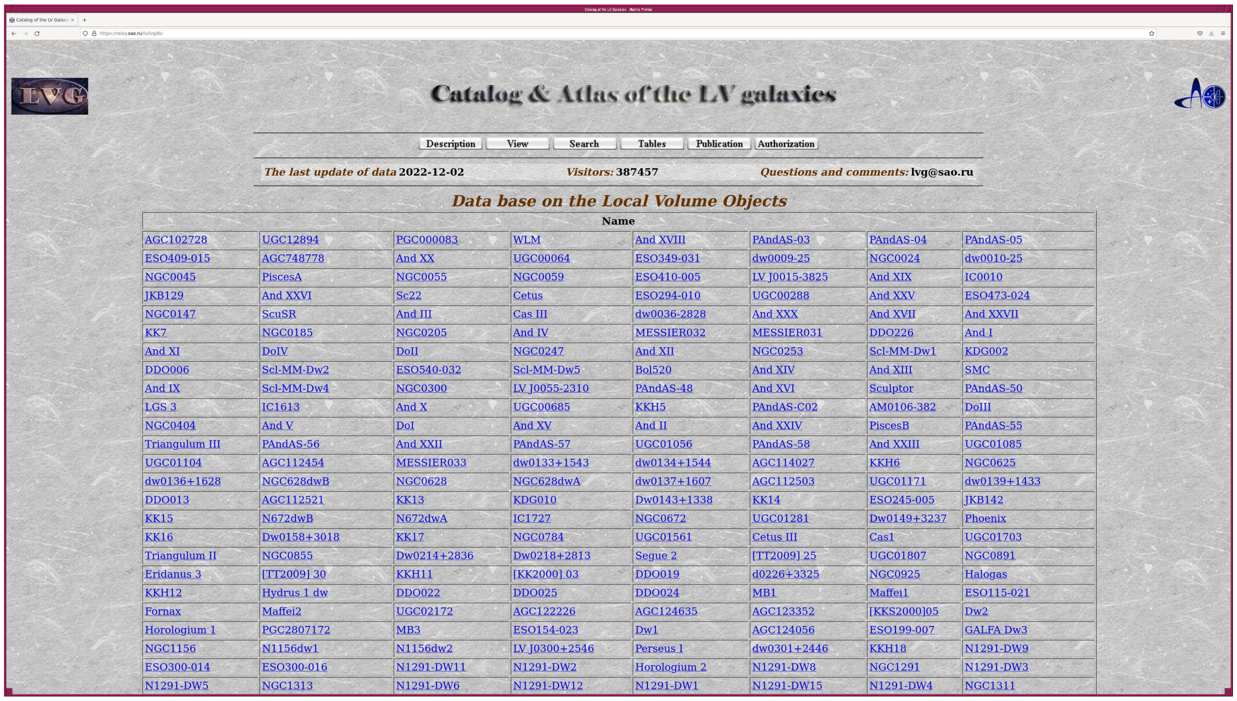
Practice 15-10

Make your own Python script to read the JSON file "`exoplanet.json`" and print the information of exoplanets hosted by TRAPPIST-1.

5 Catalog & Atlas of the LV galaxies

Visit the website of the Catalog & Atlas of the LV galaxies. (Fig. 7)

- `https://serv.sao.ru/lv/lvgdb/`



The last update of data 2022-12-02 *Visitors:* 387457 *Questions and comments:* lvg@sao.ru

Data base on the Local Volume Objects

Name							
AGC102728	UGC12894	PGC000083	WLM	And XVIII	PAndAS-03	PAndAS-04	PAndAS-05
ESO409-015	AGC748778	And XX	UGC00064	ESO349-031	dw0009-25	NGC0024	dw0010-25
NGC0045	PiscesA	NGC0055	NGC0059	ESO410-005	LV J0015-3825	And XIX	IC0010
JKB129	And XXVI	Sc22	Cetus	ESO294-010	UGC00288	And XXV	ESO473-024
NGC0147	ScuSR	And III	Cas III	dw0036-2828	And XXX	And XVII	And XXVII
KK7	NGC0185	NGC0205	And IV	MESSIER032	MESSIER031	DDO226	And I
And XI	DoIV	DoII	NGC0247	And XII	NGC0253	Scl-MM-Dw1	KDG002
DDO006	Scl-MM-Dw2	ESO540-032	Scl-MM-Dw5	Bol520	And XIV	And XIII	SMC
And IX	Scl-MM-Dw4	NGC0300	LV J0055-2310	PAndAS-48	And XVI	Sculptor	PAndAS-50
LGS 3	IC1613	And X	UGC00685	KKH5	PAndAS-C02	AM0106-382	DoIII
NGC0404	And V	DoI	And XV	And II	And XXIV	PiscesB	PAndAS-55
Triangulum III	PAndAS-56	And XXII	PAndAS-57	UGC01056	PAndAS-58	And XXIII	UGC01085
UGC01104	AGC112454	MESSIER033	dw0133+1543	dw0134+1544	AGC114027	KKH6	NGC0625
dw0136+1628	NGC628dwB	NGC0628	NGC628dwA	dw0137+1607	AGC112503	UGC01171	dw0139+1433
DDO013	AGC112521	KK13	KDG010	Dw0143+1338	KK14	ESO245-005	JKB142
KK15	N672dwB	N672dwA	IC1727	NGC0672	UGC01281	Dw0149+3237	Phoenix
KK16	Dw0158+3018	KK17	NGC0784	UGC01561	Cetus III	Cas1	UGC01703
Triangulum II	NGC0855	Dw0214+2836	Dw0218+2813	Segue 2	[TT2009] 25	UGC01807	NGC0891
Eridanus 3	[TT2009] 30	KKH11	[KK2000] 03	DDO019	d0226+3325	NGC0925	Halogas
KKH12	Hydrus 1 dw	DDO022	DDO025	DDO024	MB1	Maffei1	ESO115-021
Fornax	Maffei2	UGC02172	AGC122226	AGC124635	AGC123352	[KKS2000]05	Dw2
Horologium 1	PGC2807172	MB3	ESO154-023	Dw1	AGC124056	ESO199-007	GALFA Dw3
NGC1156	N1156dw1	N1156dw2	LV J0300+2546	Perseus I	dw0301+2446	KKH18	N1291-DW9
ESO300-014	ESO300-016	N1291-DW11	N1291-DW2	Horologium 2	N1291-DW8	NGC1291	N1291-DW3
N1291-DW5	NGC1313	N1291-DW6	N1291-DW12	N1291-DW1	N1291-DW15	N1291-DW4	NGC1311

Figure 7: The official website of the Catalog & Atlas of the LV galaxies.

5.1 Downloading velocity table

Make a Python script to download the velocity table of Catalog & Atlas of the LV galaxies. Here is an example.

Python Code 11: ai202209_s15_03_00.py

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

#
# Time-stamp: <2022/12/25 20:13:42 (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://relay.sao.ru/lv/lvgdb/tables/lvg_table4.dat'

# output file name
file_output = 'lvg_v.data'

# 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 velocity table.

```
% chmod a+x ai202209_s15_03_00.py
% ./ai202209_s15_03_00.py
Now, fetching https://relay.sao.ru/lv/lvgdb/tables/lvg_table4.dat...
Finished fetching https://relay.sao.ru/lv/lvgdb/tables/lvg_table4.dat!
Now, writing the data into file "lvg_v.data"...
Finished writing the data into file "lvg_v.data"!
% ls -lF lvg*.data
-rw-r--r--  1 daisuke  taiwan  47660 Dec 25 20:15 lvg_v.data
% head -20 lvg_v.data
Title: Catalog&Atlas of the LV galaxies (LVG)
Table: List of heliocentric velocities
================================================================================
Byte-by-byte Description of file: lvg_table4.dat
--------------------------------------------------------------------------------
   Bytes Format Units   Label  Explanations
--------------------------------------------------------------------------------
   1- 18 A18    ---     Name   Galaxy name in well-known catalogs
  20- 23 I4     km/s    cz     Heliocentric velocity
  25- 27 I3     km/s  e_cz     ? Error in cz
  29- 47 A19    ---   r_cz     Reference for cz
--------------------------------------------------------------------------------
AGC102728           566   4 2011AJ....142..170H
UGC12894            335  22 1992ApJS...81....5S
PGC000083           542   8 2018ApJ...861...49H
WLM                -122   2 2004AJ....128...16K
And XVIII          -332   3 2012ApJ...752...45T
PAndAS-04          -397   7 2014MNRAS.442.2165H
PAndAS-05          -183   7 2014MNRAS.442.2165H
ESO409-015          726  18 2005MNRAS.361...34D
```

5.2 Downloading distance table

Make a Python script to download the distance table of Catalog & Atlas of the LV galaxies. Here is an example.

Python Code 12: ai202209_s15_03_01.py

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

#
# Time-stamp: <2022/12/25 20:13:50 (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://relay.sao.ru/lv/lvgdb/tables/lvg_table6.dat'

# output file name
file_output = 'lvg_d.data'

# 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 velocity table.

```
% chmod a+x ai202209_s15_03_01.py
% ./ai202209_s15_03_01.py
Now, fetching https://relay.sao.ru/lv/lvgdb/tables/lvg_table6.dat...
Finished fetching https://relay.sao.ru/lv/lvgdb/tables/lvg_table6.dat!
Now, writing the data into file "lvg_d.data"...
Finished writing the data into file "lvg_d.data"!
% ls -lF lvg*.data
-rw-r--r--  1 daisuke  taiwan  80037 Dec 25 20:16 lvg_d.data
-rw-r--r--  1 daisuke  taiwan  47660 Dec 25 20:15 lvg_v.data
% head -20 lvg_d.data
```

```
Title: Catalog&Atlas of the LV galaxies (LVG)
Table: List of distances
================================================================================
Byte-by-byte Description of file: lvg_table6.dat
--------------------------------------------------------------------------------
   Bytes Format Units   Label  Explanations
--------------------------------------------------------------------------------
   1- 18 A18    ---     Name   Galaxy name in well-known catalogs
  20- 24 F5.2   mag     DM     Distance modulus
  26- 29 F4.2   mag   e_DM     ? Error in DM
  31- 34 A4     ---   n_DM     Method used to determine DM (1)
  36- 54 A19    ---   r_DM     Reference for DM
--------------------------------------------------------------------------------
Note (1): References to the distance estimation are presented in this Table and in LVG.
    TRGB = by the tip of the red giant branch;
     Cep = from the Cepheid luminosity;
    geom = by a geometric determination of the distance;
      SN = from the Supernova luminosity;
     SBF = from galaxy surface brightness fluctuations;
     mem = from galaxy membership in known groups with measured distances of
```

5.3 Reading tables

Read velocity and distance tables, and find galaxies having both velocity and distance.

Python Code 13: ai202209_s15_03_02.py

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

#
# Time-stamp: <2022/12/25 20:32:35 (CST) daisuke>
#

# data files (distance table and velocity table)
file_d = 'lvg_d.data'
file_v = 'lvg_v.data'

# output file name
file_output = 'lvg.data'

# dictionary to store data
data_d = {}
data_v = {}

# opening file
with open (file_d, 'r') as fh_d:
    # reading distance table
    counter = 0
    for line in fh_d:
        # increment the counter if the line starts with '-'.
        if (line[0] == '-'):
            counter += 1
        # stop processing the line if the counter is smaller than 4.
        if ( (counter < 4) or (line[0] == '-') ):
            continue
        # extracting data
        name_str     = line[0:18].strip ()
        dm_str       = line[19:24].strip ()
        dm_err_str   = line[25:29].strip ()
        d_method_str = line[30:34].strip ()
        # conversion from string to float
        dm = float (dm_str)
        if (dm_err_str == ''):
            dm_err = 0.0
        else:
            dm_err = float (dm_err_str)
        # calculation of distance in Mpc
        dist_pc      = 10**(dm / 5.0 + 1.0)
        dist_Mpc     = dist_pc * 10**-6
        dist_pc_err  = 10**( (dm + dm_err) / 5.0 + 1.0 ) - dist_pc
        dist_Mpc_err = dist_pc_err * 10**-6
        # constructing dictionary
        data_d[name_str]                  = {}
        data_d[name_str]['dm']            = dm
        data_d[name_str]['dm_err']        = dm_err
        data_d[name_str]['dist_Mpc']      = dist_Mpc
        data_d[name_str]['dist_Mpc_err'] = dist_Mpc_err
        data_d[name_str]['method']        = d_method_str

# opening file
with open (file_v, 'r') as fh_v:
    # reading velocity table
    counter = 0
    for line in fh_v:
        # increment the counter if the line starts with '-'.
        if (line[0] == '-'):
            counter += 1
        # stop processing the line if the counter is smaller than 4.
        if ( (counter < 3) or (line[0] == '-') ):
            continue
        # extracting data
        name_str    = line[0:18].strip ()
        vel_str     = line[19:23].strip ()
        vel_err_str = line[24:27].strip ()
        # conversion from string to float
        vel = float (vel_str)
        if (vel_err_str == ''):
            vel_err = 0.0
        else:
            vel_err = float (vel_err_str)
        # constructing dictionary
        data_v[name_str] = {}
        data_v[name_str]['vel'] = vel
        data_v[name_str]['vel_err'] = vel_err

with open (file_output, 'w') as fh_out:
    # printing the header
    header = f"# LVG galaxies with known distance and velocity\n" \
        + f"# distance in Mpc, distance error in Mpc,\n" \
        + f"# velocity in km/s, velocity error in km/s,\n" \
        + f"# method of distance determination, name of galaxy\n"
    fh_out.write (header)

    # finding galaxies with both known distance and velocity
    # sorting the data by distance
    for name in sorted (data_d, key=lambda x: data_d[x]['dist_Mpc']):
        # if a galaxy has velocity data, then we print the data.
        if name in data_v:
            # printing data
            record = f"{data_d[name]['dist_Mpc']:12.6f}" \
                + f" {data_d[name]['dist_Mpc_err']:12.6f}" \
                + f" {data_v[name]['vel']:12.6f}" \
                + f" {data_v[name]['vel_err']:12.6f}" \
                + f" # {data_d[name]['method']}, {name}\n"
            fh_out.write (record)
```

Execute above script to find galaxies having both velocity and distance measurements and write the data into a file.

```
% chmod a+x ai202209_s15_03_02.py
% ./ai202209_s15_03_02.py
% ls -lF lvg*.data
-rw-r--r--  1 daisuke  taiwan  66871 Dec 25 20:32 lvg.data
-rw-r--r--  1 daisuke  taiwan  80037 Dec 25 20:16 lvg_d.data
-rw-r--r--  1 daisuke  taiwan  47660 Dec 25 20:15 lvg_v.data
% head lvg.data
# LVG galaxies with known distance and velocity
# distance in Mpc, distance error in Mpc,
# velocity in km/s, velocity error in km/s,
# method of distance determination, name of galaxy
    0.007943     0.000413     0.000000     0.000000 # geom, Milky Way
    0.020045     0.000000   140.000000     2.000000 # TRGB, Sag dSph
    0.022909     0.002210   206.000000     1.000000 # CMD, Segue 1
    0.025003     0.002036  -102.000000     2.000000 # HB, Tucana III
    0.027542     0.007131    80.000000     1.000000 # HB, Hydrus 1 dw
    0.027797     0.001310   285.000000     3.000000 # HB, Carina III
```

Try following practice.

Practice 15-11

Make your own Python script to find having both velocity and distance measurements.

5.4 Making a Hubble diagram for LV galaxies

Make a Python script to generate a Hubble diagram for nearby galaxies using LV galaxies catalogue. Here is an example.

Python Code 14: ai202209_s15_03_03.py

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

#
# Time-stamp: <2022/12/25 20:31:36 (CST) daisuke>
#

# importing argparse module
import argparse

# importing numpy module
import numpy

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

# command-line argument analysis
desc = 'making Hubble diagram using LVG data'
```

```
parser = argparse.ArgumentParser (description=desc)
parser.add_argument ('-i', '--input', help='input data file name')
parser.add_argument ('-o', '--output', help='output figure file name')
args = parser.parse_args ()

# file names
file_data = args.input
file_fig  = args.output

# numpy arrays to store data
data_d     = numpy.array ([])
data_d_err = numpy.array ([])
data_v     = numpy.array ([])
data_v_err = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # reading data
    for line in fh:
        # skip the line, if it starts with '#'
        if (line[0] == '#'):
            continue
        # splitting the line
        data  = line.split ()
        # extracting data
        d     = float (data[0])
        d_err = float (data[1])
        v     = float (data[2])
        v_err = float (data[3])
        # appending data to numpy arrays
        # reject data with distance error larger than 10.0 Mpc
        if (d_err < 10.0):
            data_d     = numpy.append (data_d, d)
            data_d_err = numpy.append (data_d_err, d_err)
            data_v     = numpy.append (data_v, v)
            data_v_err = numpy.append (data_v_err, v_err)

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

# axes
ax.set_xlabel ('Distance [Mpc]')
ax.set_ylabel ('Velocity [km/s]')
ax.grid ()

# making a Hubble diagram
ax.errorbar (data_d, data_v, xerr=data_d_err, yerr=data_v_err, \
             linestyle='None', marker='o', color='blue', markersize=3, \
             ecolor='black', capsize=3, label='LVG galaxies')
ax.legend ()

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

Execute above script to make a Hubble diagram for nearby galaxies.

```
% chmod a+x ai202209_s15_03_03.py
% ./ai202209_s15_03_03.py -h
usage: ai202209_s15_03_03.py [-h] [-i INPUT] [-o OUTPUT]

making Hubble diagram using LVG data

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input data file name
  -o OUTPUT, --output OUTPUT
                        output figure file name

% ./ai202209_s15_03_03.py -i lvg.data -o lvg.png
% ls -lF lvg*.png
-rw-r--r--  1 daisuke  taiwan  139571 Dec 25 20:41 lvg.png
```

Display the PNG file. (Fig. 8)

```
% feh -dF lvg.png
```

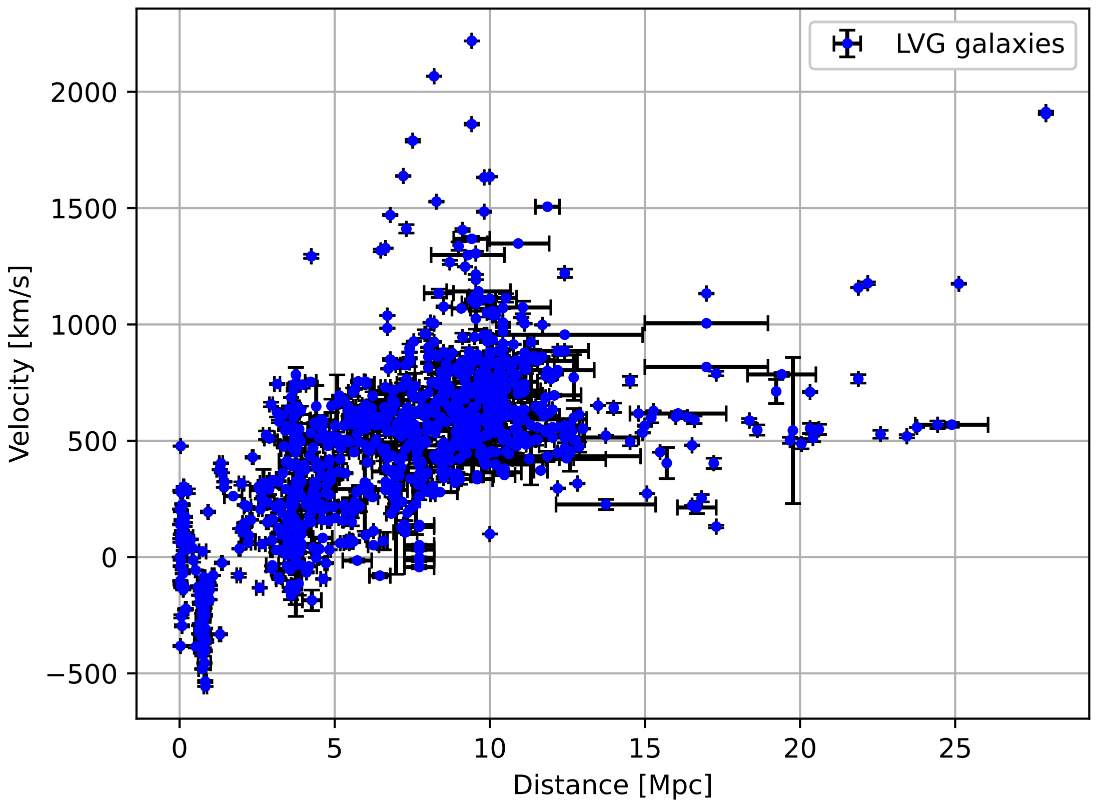


Figure 8: The Hubble diagram constructed from LVG database.

Try following practice.

Practice 15-12

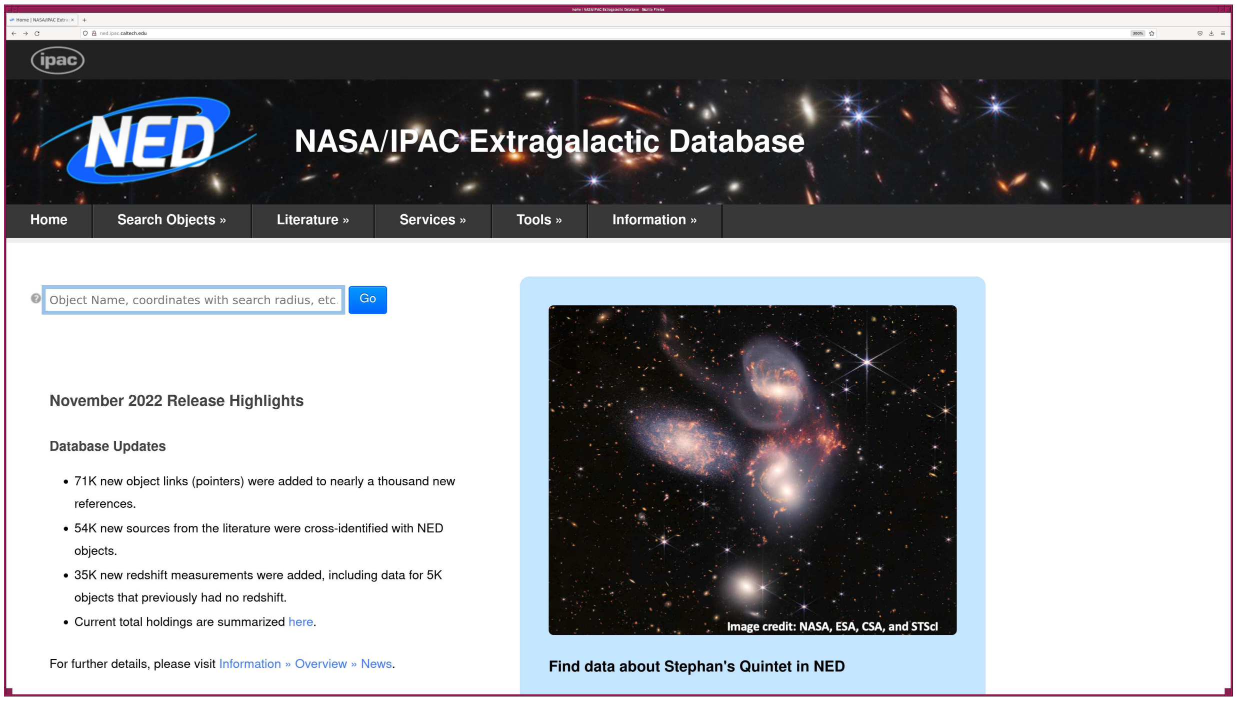
Change marker shape, size, and colour, and make the Hubble diagram again.

6 NED-1D database

Visit following web page for downloading NED-1D galaxy database.

- NASA/IPAC Extragalactic Database: `http://ned.ipac.caltech.edu/`

First, go to NED web page. (Fig. 9) Move the mouse cursor to the "Services" and show the pull-down menu. (Fig. 10) Click the menu "Level5". (Fig. 11) Click the link "Database of Galaxy Distances". (Fig. 12) Then, you find a link to the CSV file for NED Redshift-Independent Distances database. (Fig. 13) Download the CSV file.


ipac
NED
NASA/IPAC Extragalactic Database
Home
Search Objects »
Literature »
Services »
Tools »
Information »
Object Name, coordinates with search radius, etc.
Go
Image credit: NASA, ESA, CSA, and STScI

November 2022 Release Highlights

Database Updates

- 71K new object links (pointers) were added to nearly a thousand new references.
- 54K new sources from the literature were cross-identified with NED objects.
- 35K new redshift measurements were added, including data for 5K objects that previously had no redshift.
- Current total holdings are summarized here.

For further details, please visit Information » Overview » News.

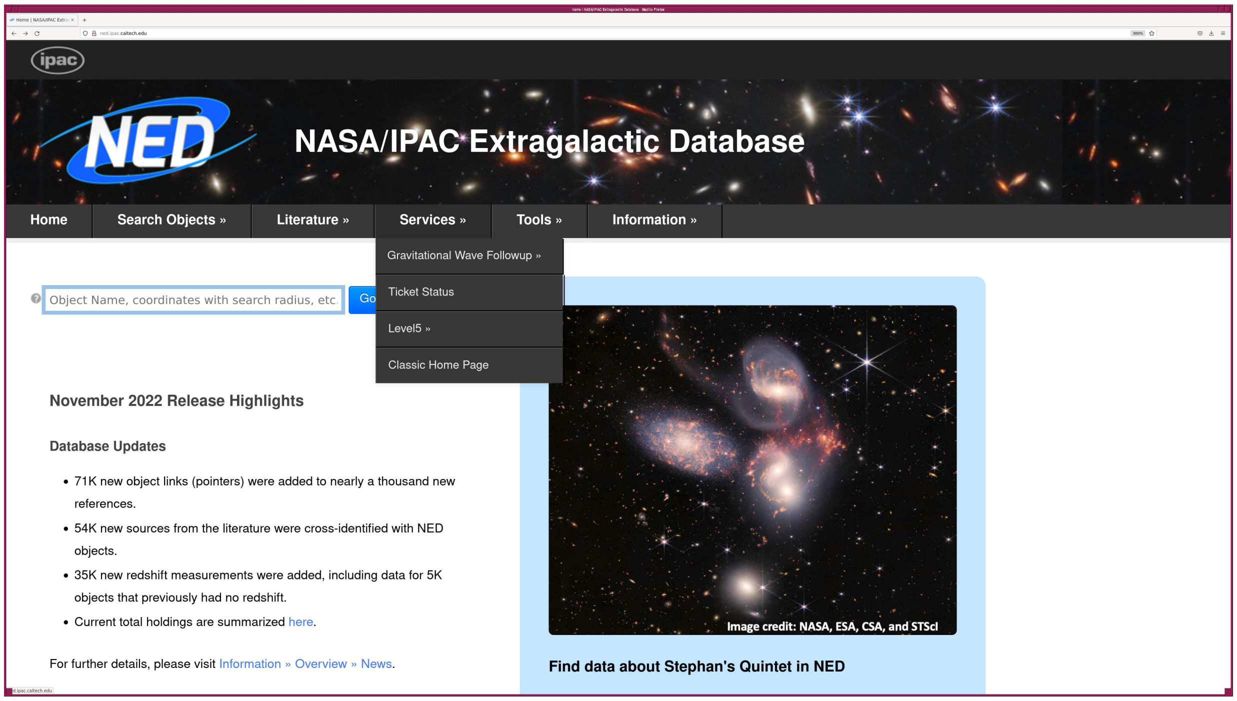
Find data about Stephan's Quintet in NED

Figure 9: The NED website.

Alternatively, try following Python script.

Python Code 15: ai202209_s15_04_00.py

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

#
# Time-stamp: <2022/12/25 21:04:37 (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://ned.ipac.caltech.edu/Archive/Distances/' \
    + 'NED30.5.1-D-17.1.2-20200415.csv'

# output file name
file_output = 'ned1d.csv'
```


ipac
NED
NASA/IPAC Extragalactic Database
Home
Search Objects »
Literature »
Services »
Tools »
Information »
Gravitational Wave Followup »
Ticket Status
Level5 »
Classic Home Page
Object Name, coordinates with search radius, etc.
Go
Image credit: NASA, ESA, CSA, and STScI

November 2022 Release Highlights

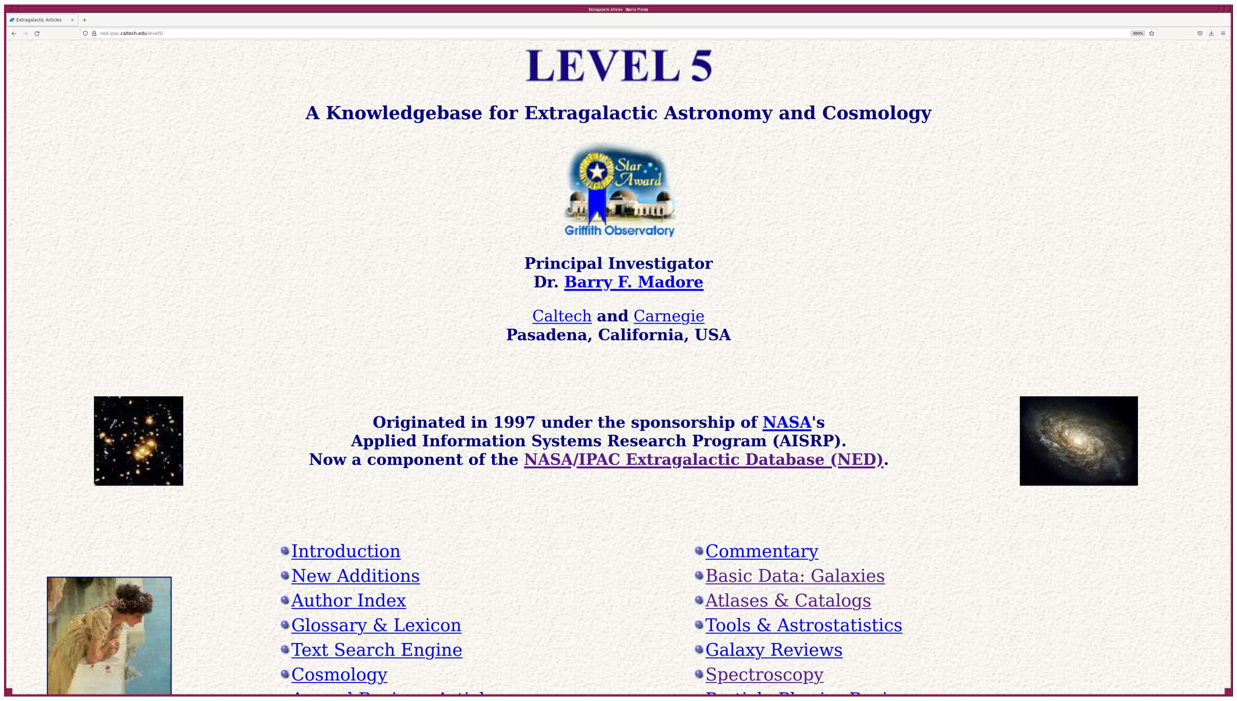
Database Updates

- 71K new object links (pointers) were added to nearly a thousand new references.
- 54K new sources from the literature were cross-identified with NED objects.
- 35K new redshift measurements were added, including data for 5K objects that previously had no redshift.
- Current total holdings are summarized here.

For further details, please visit Information » Overview » News.

Find data about Stephan's Quintet in NED

Figure 10: The Services menu of the NED website.



LEVEL 5

A Knowledgebase for Extragalactic Astronomy and Cosmology

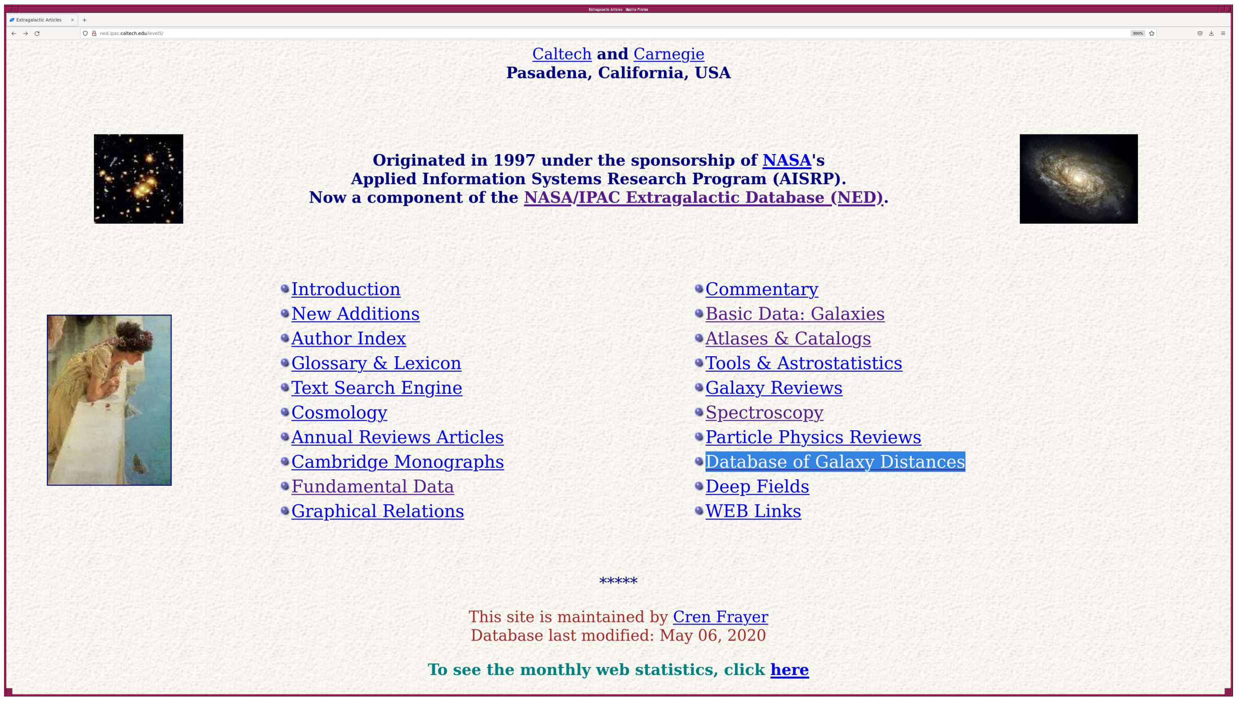
Principal Investigator
Dr. Barry F. Madore

Caltech and Carnegie
Pasadena, California, USA

Originated in 1997 under the sponsorship of NASA's
Applied Information Systems Research Program (AISRP).
Now a component of the NASA/IPAC Extragalactic Database (NED).

Introduction	Commentary
New Additions	Basic Data: Galaxies
Author Index	Atlases & Catalogs
Glossary & Lexicon	Tools & Astrostatistics
Text Search Engine	Galaxy Reviews
Cosmology	Spectroscopy

Figure 11: The Level-5 web page of the NED website.



Caltech and Carnegie
Pasadena, California, USA

Originated in 1997 under the sponsorship of NASA's
Applied Information Systems Research Program (AISRP).
Now a component of the NASA/IPAC Extragalactic Database (NED).

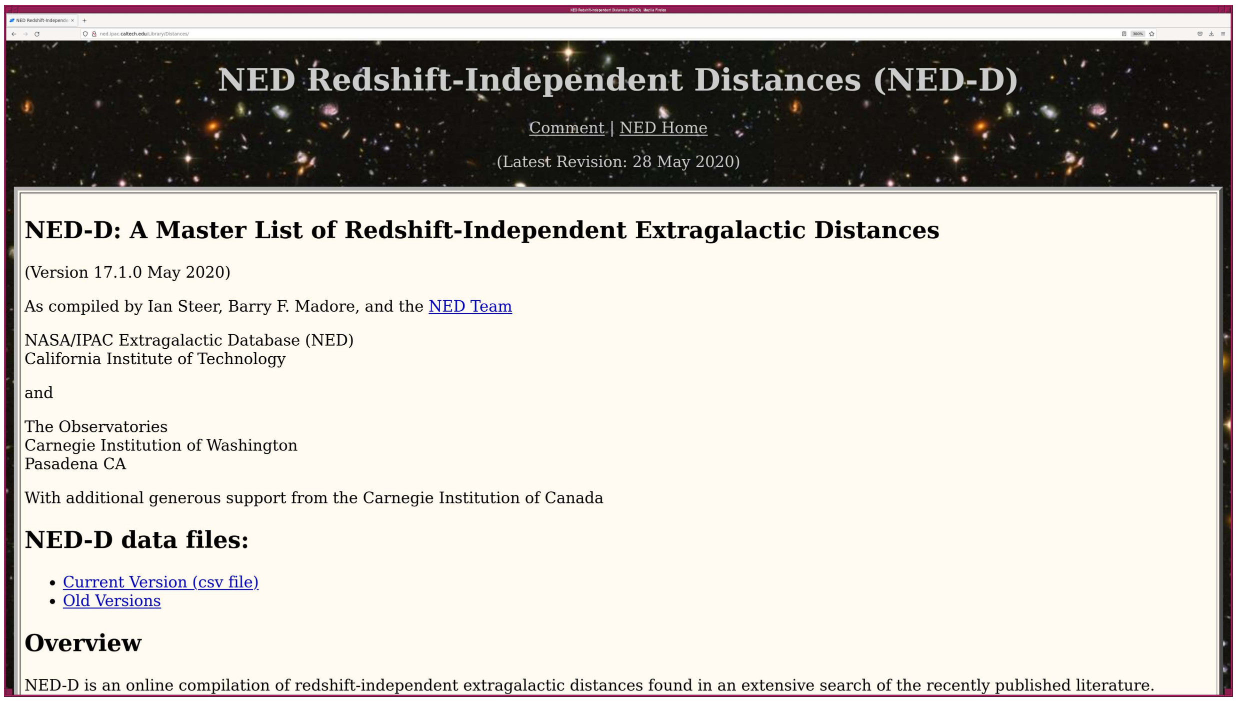
- Introduction
- New Additions
- Author Index
- Glossary & Lexicon
- Text Search Engine
- Cosmology
- Annual Reviews Articles
- Cambridge Monographs
- Fundamental Data
- Graphical Relations
- Commentary
- Basic Data: Galaxies
- Atlases & Catalogs
- Tools & Astrostatistics
- Galaxy Reviews
- Spectroscopy
- Particle Physics Reviews
- Database of Galaxy Distances
- Deep Fields
- WEB Links

*****

This site is maintained by Cren Frayer
Database last modified: May 06, 2020

To see the monthly web statistics, click here

Figure 12: The link “Database of Galaxy Distances” on the NED website.



NED Redshift-Independent Distances (NED-D)

Comment | NED Home

(Latest Revision: 28 May 2020)

NED-D: A Master List of Redshift-Independent Extragalactic Distances

(Version 17.1.0 May 2020)

As compiled by Ian Steer, Barry F. Madore, and the NED Team

NASA/IPAC Extragalactic Database (NED)
California Institute of Technology

and

The Observatories
Carnegie Institution of Washington
Pasadena CA

With additional generous support from the Carnegie Institution of Canada

NED-D data files:

- Current Version (csv file)
- Old Versions

Overview

NED-D is an online compilation of redshift-independent extragalactic distances found in an extensive search of the recently published literature.

Figure 13: The “NED Redshift-Independent Distances (NED-D)“ on the NED website.

```
# 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 NED-D data file.

```
% chmod a+x ai202209_s15_04_00.py
% ./ai202209_s15_04_00.py
Now, fetching http://ned.ipac.caltech.edu/Archive/Distances/NED30.5.1-D-17.1.2-20
200415.csv...
Finished fetching http://ned.ipac.caltech.edu/Archive/Distances/NED30.5.1-D-17.1.
2-20200415.csv!
Now, writing the data into file "ned1d.csv"...
Finished writing the data into file "ned1d.csv"!
% ls -lF ned1d.csv
-rw-r--r--  1 daisuke  taiwan  41077792 Dec 25 21:06 ned1d.csv
% head -20 ned1d.csv | cut -b 1-80
"FRN","Header Lines","13","","","","","","","","","","","",""
"FRN","Name","NED-D","","","","","","","","","","","",""
"FRN","Version","17.1.0","","","","","","","","","","","",""
"FRN","Version Date","4/15/2020","","","","","","","","","","","",""
"FRN","Compiled By","I. Steer","","","","","","","","","","","",""
"FRN","NED Version","N30.5.1","","","","","","","","","","","",""
"FRN","NED Version Date","5/11/2020","","","","","","","","","","","",""
"FRN","URI","http://ned.ipac.caltech.edu/Library/Distances/NED30.5.1-D-17.1.2-20
"FRN","Content-Type","text/csv","","","","","","","","","","","",""
"FRN","Content Format","http://tools.ietf.org/html/rfc4180","","","","","","",""
"FRN","Acknowledgement","This research has made use of the NASA/IPAC Extragalact
"FRN","Copyright","(C) 2012-2020 California Institute of Techology - All Rights
"FRN","Exclusion Code","D","G","Galaxy ID","m-M","err","D (Mpc)","Method","REFCO
"","1","1","SDSS-II SN 13651","41.64","0.17","1700","SNIa SDSS","2018PASP..130f4
"","2","1","SDSS-II SN 13651","42.1","0.21","2110","SNIa SDSS","2018PASP..130f40
"","999999","1","SDSS-II SN 13651","41.64","0.17","1700","SNIa SDSS","2018PASP..
"","999999","1","SDSS-II SN 13651","42.1","0.21","2110","SNIa SDSS","2018PASP..1
"","3","2","2MASX J00000138+1530350","38.86","0.46","592","FP","2016A&A...596A..
"","4","3","2MASX J00000155-0929403","39.79","0.46","910","FP","2016A&A...596A..
"","5","4","UGC 12889","33.65","0.66","53.7","Tully-Fisher","1984A&AS...56..381B
```

First 13 lines of the CSV file is the header. Remove the header.

```
% wc ned1d.csv
  328330 1131477 41077792 ned1d.csv
% tail -328317 ned1d.csv > ned1d_noheader.csv
% ls -lF ned1d*
-rw-r--r--  1 daisuke  taiwan  41077792 Dec 25 21:06 ned1d.csv
-rw-r--r--  1 daisuke  taiwan  41076504 Dec 25 21:16 ned1d_noheader.csv
% head -20 ned1d_noheader.csv | cut -b 1-80
"","1","1","SDSS-II SN 13651","41.64","0.17","1700","SNIa SDSS","2018PASP..130f4
"","2","1","SDSS-II SN 13651","42.1","0.21","2110","SNIa SDSS","2018PASP..130f40
"","999999","1","SDSS-II SN 13651","41.64","0.17","1700","SNIa SDSS","2018PASP..
"","999999","1","SDSS-II SN 13651","42.1","0.21","2110","SNIa SDSS","2018PASP..1
"","3","2","2MASX J00000138+1530350","38.86","0.46","592","FP","2016A&A...596A..
"","4","3","2MASX J00000155-0929403","39.79","0.46","910","FP","2016A&A...596A..
"","5","4","UGC 12889","33.65","0.66","53.7","Tully-Fisher","1984A&AS...56..381B
"","6","4","UGC 12889","34.16","0.4","68","Tully-Fisher","2007A&A...465...71T","
"","7","4","UGC 12889","34.66","0.45","85.6","Tully-Fisher","2007A&A...465...71T
"","8","4","UGC 12889","34.68","0.47","86.1","Tully-Fisher","2007A&A...465...71T
"","9","4","UGC 12889","34.7","0.46","86.9","Tully-Fisher","2007A&A...465...71T"
"","10","5","KUG 2357+228","33.52","0.2","50.6","Tully-Fisher","2013AJ....146...
"","11","5","KUG 2357+228","33.73","0.37","55.8","Tully-Fisher","2009ApJS..182..
"","12","5","KUG 2357+228","34.11","0.31","66.3","Tully-Fisher","2009ApJS..182..
"","13","6","KUG 2357+257","35.65","0.2","135","Tully-Fisher","2013AJ....146...8
"","14","7","SDSS-II SN 14480","40.19","0.3","1090","SNIa","2011ApJ...738..162S"
"","15","7","SDSS-II SN 14480","41.2","0.33","1300","SNIa","2018PASP..130f4002S"
"","16","7","SDSS-II SN 14480","41.3","0.32","1370","SNIa","2018PASP..130f4002S"
"","17","7","SDSS-II SN 14480","41.07","0.22","1640","SNIa","2013ApJ...763...88C
"","18","7","SDSS-II SN 14480","41.11","0.22","1670","SNIa","2013ApJ...763...88C
```

Generate a header for NED-D CSV file.

```
% echo '"Exclusion Code","D","G","Galaxy ID","m-M","err","D (Mpc)","Method","REFC
ODE","","","Hubble const.","Adopted LMC modulus","Date (Yr. - 1980)","NOTES"' > n
ed1d_header.csv
% cat ned1d_header.csv
"Exclusion Code","D","G","Galaxy ID","m-M","err","D (Mpc)","Method","REFCODE","",
"","Hubble const.","Adopted LMC modulus","Date (Yr. - 1980)","NOTES"
```

Concatenate the header and the data file.

```
% cat ned1d_header.csv ned1d_noheader.csv > ned1d_with_header.csv
% ls -lF ned1d*
-rw-r--r--  1 daisuke  taiwan  41077792 Dec 25 21:06 ned1d.csv
-rw-r--r--  1 daisuke  taiwan     44132 Dec 25 21:58 ned1d.png
-rw-r--r--  1 daisuke  taiwan       150 Dec 25 22:02 ned1d_header.csv
-rw-r--r--  1 daisuke  taiwan  41076504 Dec 25 21:21 ned1d_noheader.csv
-rw-r--r--  1 daisuke  taiwan  41076654 Dec 25 22:03 ned1d_with_header.csv
% head -20 ned1d_with_header.csv | cut -b 1-80
"Exclusion Code","D","G","Galaxy ID","m-M","err","D (Mpc)","Method","REFCODE",""
"","1","1","SDSS-II SN 13651","41.64","0.17","1700","SNIa SDSS","2018PASP..130f4
"","2","1","SDSS-II SN 13651","42.1","0.21","2110","SNIa SDSS","2018PASP..130f40
"","999999","1","SDSS-II SN 13651","41.64","0.17","1700","SNIa SDSS","2018PASP..
"","999999","1","SDSS-II SN 13651","42.1","0.21","2110","SNIa SDSS","2018PASP..1
"","3","2","2MASX J00000138+1530350","38.86","0.46","592","FP","2016A&A...596A..
"","4","3","2MASX J00000155-0929403","39.79","0.46","910","FP","2016A&A...596A..
"","5","4","UGC 12889","33.65","0.66","53.7","Tully-Fisher","1984A&AS...56..381B
"","6","4","UGC 12889","34.16","0.4","68","Tully-Fisher","2007A&A...465...71T","
```

```
"","7","4","UGC 12889","34.66","0.45","85.6","Tully-Fisher","2007A&A...465...71T
"","8","4","UGC 12889","34.68","0.47","86.1","Tully-Fisher","2007A&A...465...71T
"","9","4","UGC 12889","34.7","0.46","86.9","Tully-Fisher","2007A&A...465...71T"
"","10","5","KUG 2357+228","33.52","0.2","50.6","Tully-Fisher","2013AJ....146...
"","11","5","KUG 2357+228","33.73","0.37","55.8","Tully-Fisher","2009ApJS..182..
"","12","5","KUG 2357+228","34.11","0.31","66.3","Tully-Fisher","2009ApJS..182..
"","13","6","KUG 2357+257","35.65","0.2","135","Tully-Fisher","2013AJ....146...8
"","14","7","SDSS-II SN 14480","40.19","0.3","1090","SNIa","2011ApJ...738..162S"
"","15","7","SDSS-II SN 14480","41.2","0.33","1300","SNIa","2018PASP..130f4002S"
"","16","7","SDSS-II SN 14480","41.3","0.32","1370","SNIa","2018PASP..130f4002S"
"","17","7","SDSS-II SN 14480","41.07","0.22","1640","SNIa","2013ApJ...763...88C"
```

6.1 Reading CSV file using astropy.io.ascii

Use `astropy.io.ascii` to read the CSV file. Here is an example.

Python Code 16: ai202209_s15_04_01.py

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

#
# Time-stamp: <2022/12/25 21:33:37 (CST) daisuke>
#

# importing astropy module
import astropy.io.ascii

# file
file_data = 'ned1d_with_header.csv'

# reading CSV data
rawdata = astropy.io.ascii.read (file_data, format='csv')

# printing astropy table summary information
print (rawdata.info ())
```

Execute above script to read the CSV file and print the summary information of Astropy table.

```
% chmod a+x ai202209_s15_04_01.py
% ./ai202209_s15_04_01.py
<Table length=328317>
        name          dtype      class      n_bad
------------------- ------- ------------ ------
     Exclusion Code    str2 MaskedColumn 257104
                  D   int64       Column      0
                  G   int64       Column      0
         Galaxy ID   str30       Column      0
                m-M float64       Column      0
                err float64 MaskedColumn  13299
            D (Mpc) float64       Column      0
             Method   str26       Column      0
            REFCODE   str19       Column      0
               col9   str27 MaskedColumn 288753
                 _1 float64 MaskedColumn 297459
      Hubble const. float64 MaskedColumn 118084
Adopted LMC modulus float64 MaskedColumn 327792
  Date (Yr. - 1980)   int64 MaskedColumn      2
              NOTES   str80 MaskedColumn 150538
None
```

Try following practice.

Practice 15-13

Make your own Python script to read the CSV file and print column names of the Astropy table.

6.2 Making a Hubble diagram using NED-1D data

Make a Python script to extract distance and redshift of galaxies, and make a Hubble diagram.

Python Code 17: ai202209_s15_04_02.py

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

#
# Time-stamp: <2022/12/25 22:14:09 (CST) daisuke>
#

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy.io.ascii
import astropy.units
import astropy.constants

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

# command-line argument analysis
desc = 'making Hubble diagram using NED-D data'
parser = argparse.ArgumentParser (description=desc)
parser.add_argument ('-i', '--input', help='input data file name')
parser.add_argument ('-o', '--output', help='output figure file name')
args = parser.parse_args ()

# file names
file_data = args.input
file_fig  = args.output

# speed of light
c = astropy.constants.c

# units
unit_m_per_s = astropy.units.m / astropy.units.s

# reading CSV data
rawdata = astropy.io.ascii.read (file_data, format='csv')

# dictionary to store data
data = {}

# examining the data
for i in range ( len (rawdata) ):
    # extracting data
    # name of galaxy
    name = rawdata[i]['Galaxy ID']
    # distance modulus
    distmod = rawdata[i]['m-M']
    # error of distance modulus
    distmod_err = rawdata[i]['err']
    # distance in Mpc
    dist_Mpc = rawdata[i]['D (Mpc)']
    # redshift
    z = rawdata[i]['_1']
    # velocity
    v = ( (z + 1.0)**2 - 1.0 ) / ( (z + 1.0)**2 + 1.0 ) * c

    # appending data to the dictionary
    # skip if redshift is missing.
    # skip if error of distance modulus is large.
    # skip if distance is larger than 500 Mpc
    # skip if velocity is larger than 50000 km/s
    if ( (z > 0.0) and (distmod_err < 0.05) and (dist_Mpc < 500.0) \
         and (v / unit_m_per_s < 5*10**7) ):
        # if the data is not in the dictionary, add data
        if not name in data:
            data[name] = {}
            data[name]['distmod'] = distmod
            data[name]['distmod_err'] = distmod_err
            data[name]['dist_Mpc'] = dist_Mpc
            data[name]['z'] = z
            data[name]['v'] = v

# making numpy arrays for plotting data
data_d = numpy.array ([])
data_v = numpy.array ([])
for name in sorted (data, key=lambda x: data[x]['dist_Mpc']):
    data_d = numpy.append (data_d, data[name]['dist_Mpc'])
    data_v = numpy.append (data_v, data[name]['v'] * 10**-3 / unit_m_per_s)

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

# axes
ax.set_xlabel ('Distance [Mpc]')
ax.set_ylabel ('Velocity [km/s]')
ax.grid ()

# making a Hubble diagram
ax.plot (data_d, data_v, \
         linestyle='None', marker='o', color='blue', markersize=2, \
         label='galaxies in NED-1D')
ax.legend ()

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

Execute above script to make a Hubble diagram.

```
% chmod a+x ai202209_s15_04_02.py
% ./ai202209_s15_04_02.py -h
usage: ai202209_s15_04_02.py [-h] [-i INPUT] [-o OUTPUT]
```

```
making Hubble diagram using NED-D data

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input data file name
  -o OUTPUT, --output OUTPUT
                        output figure file name

% ./ai202209_s15_04_02.py -i ned1d_with_header.csv -o ned1d.png
% ls -lF ned1d.png
-rw-r--r--  1 daisuke  taiwan  76259 Dec 25 22:16 ned1d.png
```

Display the PNG file. (Fig. 14)

```
% feh -dF ned1d.png
```

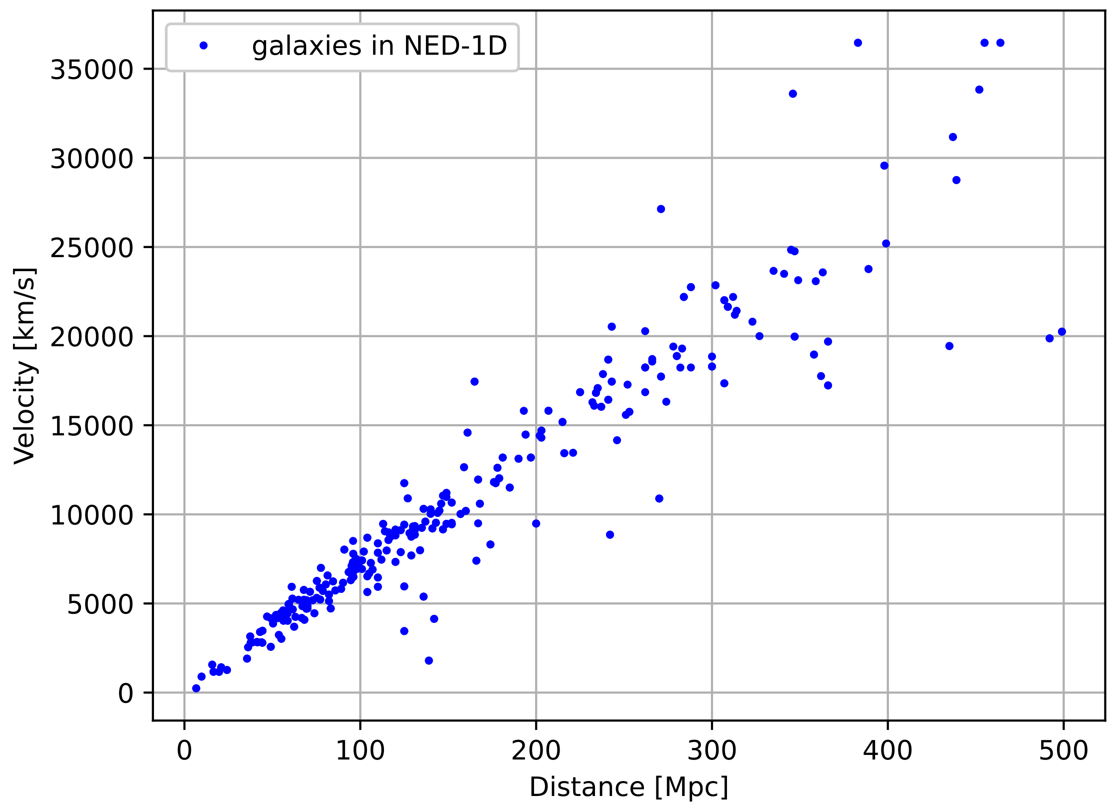


Figure 14: The Hubble diagram constructed from NED-D database.

Try following practice.

Practice 15-14

Change marker shape, size, and colour, and make the Hubble diagram again.

6.3 Determination of Hubble constant

Carry out the least-squares method to determine Hubble constant using NED-1D data.

Python Code 18: ai202209_s15_04_03.py

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

#
# Time-stamp: <2022/12/25 22:30:20 (CST) daisuke>
#

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing scipy module
import scipy.optimize
import scipy.constants

# importing astropy module
import astropy.io.ascii
import astropy.units
import astropy.constants

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

# command-line argument analysis
desc = 'making Hubble diagram using NED-D data'
parser = argparse.ArgumentParser (description=desc)
parser.add_argument ('-i', '--input', help='input data file name')
parser.add_argument ('-o', '--output', help='output figure file name')
args = parser.parse_args ()

# file names
file_data = args.input
file_fig  = args.output

# speed of light
c = astropy.constants.c

# units
unit_m_per_s = astropy.units.m / astropy.units.s

# reading CSV data
rawdata = astropy.io.ascii.read (file_data, format='csv')

# dictionary to store data
data = {}

# examining the data
for i in range ( len (rawdata) ):
    # extracting data
    # name of galaxy
    name = rawdata[i]['Galaxy ID']
    # distance modulus
    distmod = rawdata[i]['m-M']
    # error of distance modulus
    distmod_err = rawdata[i]['err']
    # distance in Mpc
    dist_Mpc = rawdata[i]['D (Mpc)']
    # redshift
    z = rawdata[i]['_1']
    # velocity
    v = ( (z + 1.0)**2 - 1.0 ) / ( (z + 1.0)**2 + 1.0 ) * c

    # appending data to the dictionary
    # skip if redshift is missing.
    # skip if error of distance modulus is large.
    # skip if distance is larger than 500 Mpc
    # skip if velocity is larger than 50000 km/s
    if ( (z > 0.0) and (distmod_err < 0.05) and (dist_Mpc < 500.0) \
         and (v / unit_m_per_s < 5*10**7) ):
        # if the data is not in the dictionary, add data
        if not name in data:
            data[name] = {}
            data[name]['distmod'] = distmod
            data[name]['distmod_err'] = distmod_err
            data[name]['dist_Mpc'] = dist_Mpc
            data[name]['z'] = z
            data[name]['v'] = v

# making numpy arrays for plotting data
data_d = numpy.array ([])
data_v = numpy.array ([])
for name in sorted (data, key=lambda x: data[x]['dist_Mpc']):
    data_d = numpy.append (data_d, data[name]['dist_Mpc'])
    data_v = numpy.append (data_v, data[name]['v'] * 10**-3 / unit_m_per_s)

# initial values of coefficient
H0 = 100.0
init = [H0]

# function
def func (x, H0):
    y = H0 * x
    return (y)

# least-squares method
popt, pcov = scipy.optimize.curve_fit (func, data_d, data_v, p0=init)

# result of fitting
H0_bestfit = popt[0]
print ("popt:")
print (popt)
print ("pcov:")
print (pcov)

# degree of freedom
dof = len (data_d) - len (init)
print (f"dof = {dof}")

# residual
residual = data_v - func (data_d, popt[0])
reduced_chi2 = (residual**2).sum () / dof
print (f"reduced chi^2 = {reduced_chi2}")
```

```
# error of H0
H0_err = numpy.sqrt (pcov[0][0])
print (f"H0 = {H0_bestfit} +/- {H0_err} ({H0_err / H0_bestfit * 100.0} %)")

# fitted curve
fitted_x = numpy.linspace (0.0, 500.0, 1000)
fitted_y = func (fitted_x, H0_bestfit)

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

# axes
ax.set_xlabel ('Distance [Mpc]')
ax.set_ylabel ('Velocity [km/s]')
ax.grid ()

# making a Hubble diagram
label_fitting = f"best-fit line (H0={H0_bestfit:4.1f})"
ax.plot (fitted_x, fitted_y, linestyle='--', linewidth=3, color='red', \
         label=label_fitting)
ax.plot (data_d, data_v, \
         linestyle='None', marker='o', color='blue', markersize=2, \
         label='galaxies in NED-1D')
ax.legend ()

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

Execute above script to determine Hubble constant using least-squares method.

```
% chmod a+x ai202209_s15_04_03.py
% ./ai202209_s15_04_03.py -h
usage: ai202209_s15_04_03.py [-h] [-i INPUT] [-o OUTPUT]

making Hubble diagram using NED-D data

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input data file name
  -o OUTPUT, --output OUTPUT
                        output figure file name

% ./ai202209_s15_04_03.py -i ned1d_with_header.csv -o ned1d_fit.png
popt:
[66.55913513]
pcov:
[[0.42119364]]
dof = 453
reduced chi^2 = 6745402.045932673
H0 = 66.55913513339581 +/- 0.6489943321224565 (0.9750642504920798 %)
% ls -lF ned1d_fit.png
-rw-r--r--  1 daisuke  taiwan  91776 Dec 25 22:27 ned1d_fit.png
```

The obtained best-fit Hubble constant is 66.56 ± 0.65 km s^{-1} Mpc^{-1}. Display the PNG file. (Fig. 15)

```
% feh -dF ned1d_fit.png
```

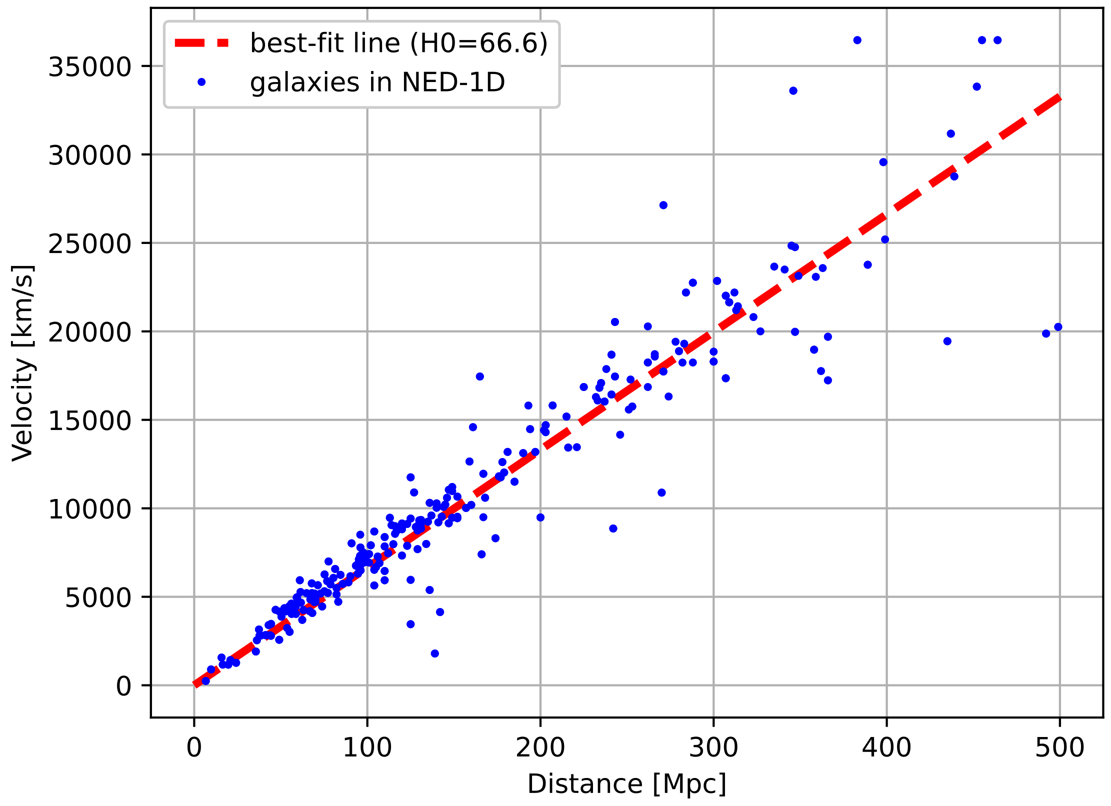


Figure 15: The Hubble diagram constructed from NED-D database and Hubble constant determined by least-squares method.

Try following practice.

Practice 15-15

Change the initial value, and carry out the fitting again.

7 Type-Ia Supernovae from Open Supernova Catalog

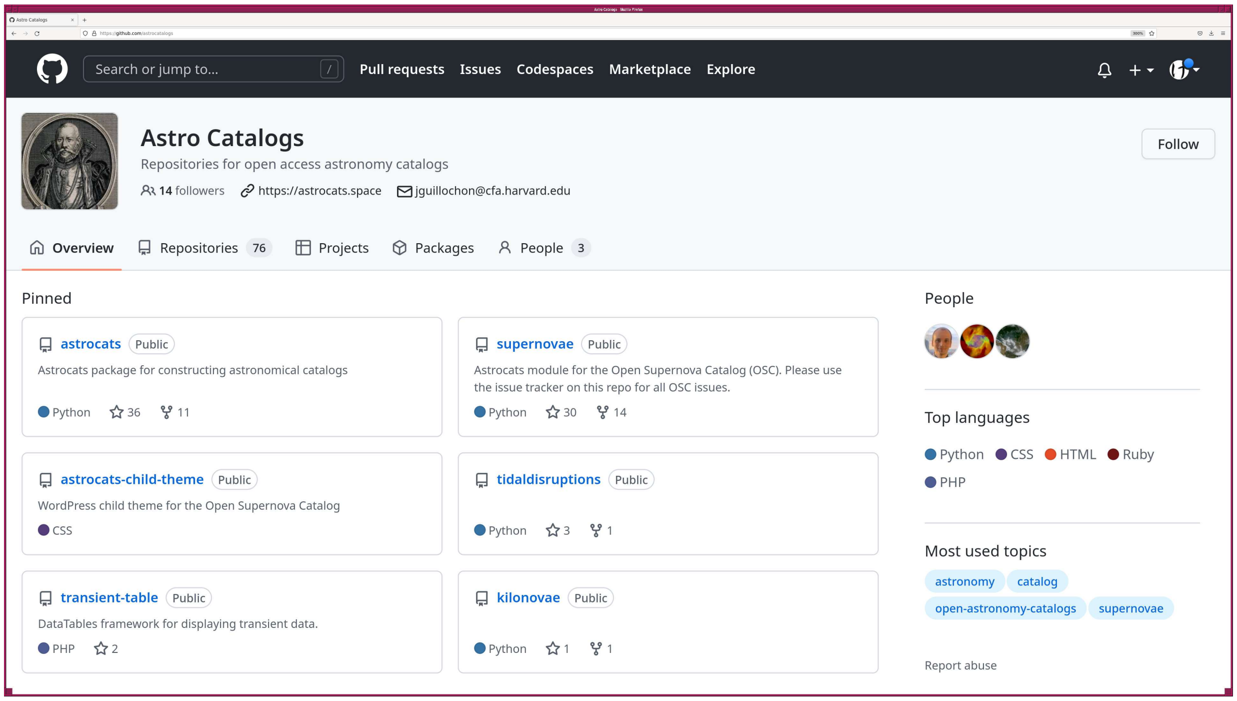
7.1 Open Supernova Catalog

The Open Supernova Catalog is available at following web page. (Fig. 16)

- `https://github.com/astrocatalogs`

7.2 Downloading Open Supernova Catalog

Use the command “`git`” to download Open Supernova Catalog.



Astro Catalogs

Repositories for open access astronomy catalogs

14 followers https://astrocats.space jguillochon@cfa.harvard.edu

Overview Repositories 76 Projects Packages People 3

Pinned

astrocats Public
Astrocats package for constructing astronomical catalogs
Python 36 11

supernovae Public
Astrocats module for the Open Supernova Catalog (OSC). Please use the issue tracker on this repo for all OSC issues.
Python 30 14

astrocats-child-theme Public
WordPress child theme for the Open Supernova Catalog
CSS

tidaldisruptions Public
Python 3 1

transient-table Public
DataTables framework for displaying transient data.
PHP 2

kilonovae Public
Python 1 1

People

Top languages
Python CSS HTML Ruby PHP

Most used topics
astronomy catalog open-astronomy-catalogs supernovae

Report abuse

Figure 16: The official website of the Open Supernova Catalog.

```
% git -c http.sslVerify=false clone \
? https://github.com/astrocatalogs/sne-pre-1990.git
Cloning into 'sne-pre-1990'...
remote: Enumerating objects: 183348, done.
remote: Counting objects: 100% (9747/9747), done.
remote: Compressing objects: 100% (3122/3122), done.
Receiving objects: 100% (183348/183348), 1.07 GiB | 3.94 MiB/s, done.
remote: Total 183348 (delta 9199), reused 6954 (delta 6625), pack-reused 173601
Resolving deltas: 100% (179416/179416), done.
Updating files: 100% (32820/32820), done.
% git -c http.sslVerify=false clone \
? https://github.com/astrocatalogs/sne-1990-1999.git
Cloning into 'sne-1990-1999'...
remote: Enumerating objects: 70279, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 70279 (delta 14), reused 10 (delta 5), pack-reused 70260
Receiving objects: 100% (70279/70279), 2.81 GiB | 3.95 MiB/s, done.
Resolving deltas: 100% (69608/69608), done.
Updating files: 100% (2349/2349), done.
% git -c http.sslVerify=false clone \
? https://github.com/astrocatalogs/sne-2000-2004.git
Cloning into 'sne-2000-2004'...
remote: Enumerating objects: 100535, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 100535 (delta 35), reused 10 (delta 8), pack-reused 100493
Receiving objects: 100% (100535/100535), 2.93 GiB | 3.79 MiB/s, done.
Resolving deltas: 100% (99676/99676), done.
Updating files: 100% (1926/1926), done.
% git -c http.sslVerify=false clone \
? https://github.com/astrocatalogs/sne-2005-2009.git
```

```
Cloning into 'sne-2005-2009'...
remote: Enumerating objects: 243371, done.
remote: Counting objects: 100% (287/287), done.
remote: Compressing objects: 100% (265/265), done.
Receiving objects: 100% (243371/243371), 2.62 GiB | 3.86 MiB/s, done.
remote: Total 243371 (delta 208), reused 26 (delta 22), pack-reused 243084
Resolving deltas: 100% (241824/241824), done.
Updating files: 100% (8000/8000), done.
% git -c http.sslVerify=false clone \
? https://github.com/astrocatalogs/sne-2010-2014.git
Cloning into 'sne-2010-2014'...
remote: Enumerating objects: 355468, done.
remote: Counting objects: 100% (297/297), done.
remote: Compressing objects: 100% (250/250), done.
remote: Total 355468 (delta 268), reused 66 (delta 47), pack-reused 355171
Receiving objects: 100% (355468/355468), 2.06 GiB | 3.90 MiB/s, done.
Resolving deltas: 100% (349471/349471), done.
Updating files: 100% (10381/10381), done.
% git -c http.sslVerify=false clone \
? https://github.com/astrocatalogs/sne-2015-2019.git
Cloning into 'sne-2015-2019'...
remote: Enumerating objects: 592944, done.
remote: Counting objects: 100% (6193/6193), done.
remote: Compressing objects: 100% (5160/5160), done.
Receiving objects: 100% (592944/592944), 631.40 MiB | 4.10 MiB/s, done.
remote: Total 592944 (delta 4855), reused 1547 (delta 1033), pack-reused 586751
Resolving deltas: 100% (587354/587354), done.
Updating files: 100% (27406/27406), done.
```

7.3 Finding data files

Make a Python script to find downloaded JSON files.

Python Code 19: ai202209_s15_05_00.py

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

#
# Time-stamp: <2022/12/25 23:02:28 (CST) daisuke>
#

# importing pathlib module
import pathlib

# list of data files
files = pathlib.Path ('.').glob ('sne-*/*.json')

# printing file names
for file in sorted (files):
    print (file)
```

Execute above script to find JSON files.

```
% ./ai202209_s15_05_00.py > sne_json.list
% head sne_json.list
sne-1990-1999/AT1991bm.json
sne-1990-1999/AT1992bv.json
sne-1990-1999/AT1992bw.json
sne-1990-1999/AT1999gy.json
```

```
sne-1990-1999/AT1999gz.json
sne-1990-1999/AT1999ha.json
sne-1990-1999/AT1999hb.json
sne-1990-1999/GRB 910421.json
sne-1990-1999/GRB 910423.json
sne-1990-1999/GRB 910424.json
% wc sne_json.list
   82863   98940 2485569 sne_json.list
```

Try following practice.

Practice 15-16

Make your own Python script to find all the JSON files under the currently working directory.

7.4 Reading a JSON file

Make a Python script to read a single JSON file.

Python Code 20: ai202209_s15_05_01.py

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

#
# Time-stamp: <2022/12/25 23:21:30 (CST) daisuke>
#

# importing json module
import json

# file name
file_json = 'sne-2005-2009/2MASSJ02051081-0447150.json'

# opening file
with open (file_json, 'r') as fh:
    # reading JSON data from file
    data = json.load (fh)

# printing data
for obj in data:
    print ("obj =", obj)
    for key in data[obj]:
        print ("  %-12s ==> %s" % (key, data[obj][key]) )
```

Execute above script to read a JSON file "`sne-2005-2009/2MASSJ02051081-0447150.json`" and check the data structure.

```
% chmod a+x ai202209_s15_05_01.py
% ./ai202209_s15_05_01.py | cut -b 1-80
obj = 2MASSJ02051081-0447150
  schema       ==> https://github.com/astrocatalogs/supernovae/blob/d3ef5fc/SCHE
  name         ==> 2MASSJ02051081-0447150
  sources      ==> [{'name': '2016A&A...594A..13P', 'bibcode': '2016A&A...594A..
  alias        ==> [{'value': '2MASSJ02051081-0447150', 'source': '3'}]
  claimedtype  ==> [{'value': 'II', 'source': '3'}]
  comovingdist ==> [{'value': '240', 'derived': True, 'u_value': 'Mpc', 'source'
  discoverdate ==> [{'value': '2005/10/06', 'derived': True, 'source': '2,3'}]
  lumdist      ==> [{'value': '249', 'derived': True, 'u_value': 'Mpc', 'source'
  redshift     ==> [{'value': '0.054', 'source': '3'}]
  velocity     ==> [{'value': '16000', 'u_value': 'km/s', 'source': '2'}]
  spectra      ==> [{'time': '53649.0', 'filename': '2MASSJ02051081-0447150_2005
```

Try following practice.

Practice 15-17

Make your own Python script to read a JSON file.

7.5 Finding type-Ia supernovae

Make a Python script to open all the JSON files and check the supernova type. For later analysis, we only use type-Ia supernovae. Select type-Ia supernovae.

Python Code 21: ai202209_s15_05_02.py

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

#
# Time-stamp: <2022/12/25 23:31:33 (CST) daisuke>
#

# importing pathlib module
import pathlib

# importing json module
import json

# list of data files
files = pathlib.Path ('.').glob ('sne-*/*.json')

# key names
key_type = 'claimedtype'

# processing each file
for file in sorted (files):
    # opening file
    with open (file, 'r') as fh:
        # reading JSON data from file
        data = json.load (fh)

    # checking all the objects in JSON file
    for obj in data:
        # skip if type is not known
        if not (key_type in data[obj]):
            continue
        # skip if supernova is not type-Ia
        if not (data[obj][key_type][0]['value'] == 'Ia'):
            continue
        # printing data
        print ("%s ==> %s" % (obj, data[obj][key_type][0]['value']) )
```

Execute above script to find type-Ia supernovae.

```
% chmod a+x ai202209_s15_05_02.py
% ./ai202209_s15_05_02.py > sne_Ia.list
% head sne_Ia.list
SN1990F ==> Ia
SN1990G ==> Ia
```

```
SN1990J ==> Ia
SN1990L ==> Ia
SN1990M ==> Ia
SN1990N ==> Ia
SN1990O ==> Ia
SN1990R ==> Ia
SN1990T ==> Ia
SN1990Y ==> Ia
% wc sne_Ia.list
   14686   47482  263580 sne_Ia.list
```

More than 10,000 type-Ia supernovae were found.
Try following practice.

Practice 15-18

Make your own Python script to find type-Ia supernovae.

7.6 Gathering data from JSON files

Make a Python script to scan all the JSON files, and select type-Ia supernovae with known distance and velocity.

Python Code 22: ai202209_s15_05_03.py

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

#
# Time-stamp: <2022/12/25 23:43:42 (CST) daisuke>
#

# importing pathlib module
import pathlib

# importing json module
import json

# importing scipy module
import scipy.constants

# constants
c = scipy.constants.c

# list of data files
files = pathlib.Path ('.').glob ('sne-*/*.json')

# key names
key_type       = 'claimedtype'
key_distance   = 'lumdist'
key_velocity   = 'velocity'
key_redshift   = 'redshift'
key_spectra    = 'spectra'
key_photometry = 'photometry'
key_maxabsmag  = 'maxvisualabsmag'
key_maxappmag  = 'maxvisualappmag'
key_maxband    = 'maxvisualband'

# processing each file
for file in sorted (files):
    # opening file
    with open (file, 'r') as fh:
        # reading JSON data from file
        data = json.load (fh)

    # checking all the objects in JSON file
    for obj in data:
        # skip if type is not known
        if not (key_type in data[obj]):
            continue
        # skip if supernova is not type-Ia
        if not (data[obj][key_type][0]['value'] == 'Ia'):
            continue
        # skip if distance is not known
        if not (key_distance in data[obj]):
            continue
        # skip if velocity is not known
        if not (key_velocity in data[obj]):
            continue
        # skip if redshift is not known
        if not (key_redshift in data[obj]):
            continue
        # skip if spectrum is missing
        if not (key_spectra in data[obj]):
            continue
        # skip if photometry is missing
        if not (key_photometry in data[obj]):
            continue
        # skip if max abs mag is missing
        if not (key_maxabsmag in data[obj]):
            continue
        # skip if max app mag is missing
        if not (key_maxappmag in data[obj]):
            continue
        # skip if max visual band is missing
        if not (key_maxband in data[obj]):
            continue
        # skip if max visual band is not B-band
        if not (data[obj][key_maxband][0]['value'] == 'B'):
            continue

        # distance, velocity, and redshift, ...
        distance_str  = data[obj][key_distance][0]['value']
        velocity_str  = data[obj][key_velocity][0]['value']
        redshift_str  = data[obj][key_redshift][0]['value']
        maxabsmag_str = data[obj][key_maxabsmag][0]['value']
        maxappmag_str = data[obj][key_maxappmag][0]['value']

        # conversion from string to float
        distance  = float (distance_str)
        velocity  = float (velocity_str)
        redshift  = float (redshift_str)
        maxabsmag = float (maxabsmag_str)
        maxappmag = float (maxappmag_str)

        # distance modulus
        distmod = maxappmag - maxabsmag
        dist_from_distmod = 10**(distmod / 5.0 + 1.0) * 10**-6

        # velocity
        v_from_z = ( (redshift + 1.0)**2 - 1.0 ) \
            / ( (redshift + 1.0)**2 + 1.0 ) * c * 10**-3

        # printing data
        print (f"{distance:15.8f} {dist_from_distmod:15.8f}", \
               f" {redshift:15.8f} {v_from_z:15.8f} # {obj}")
```

Execute above script to find type-Ia supernovae with known distance and velocity.

```
% chmod a+x ai202209_s15_05_03.py
% ./ai202209_s15_05_03.py > snia.data
% head snia.data
    19.70000000     19.67886290       0.00339900   1017.26279345 # SN1990N
   118.00000000    118.03206357       0.03066000   9050.79320041 # SN1990O
   177.85700000    177.86070114       0.03903400  11473.87627745 # SN1990Y
   237.00000000    236.59196975       0.05500100  16036.08158526 # SN1991S
    17.50000000     17.53880502       0.00421000   1259.46949587 # SN1992A
    27.00000000     26.42408757       0.00529000   1581.70745014 # SN1992G
    90.00000000    106.65961212       0.02000000   5935.90242266 # SN1993Y
    26.00000000     26.54605562       0.00452000   1351.99950139 # SN1993Z
   204.00000000    204.17379447       0.04430000  12986.91218730 # SN1993ac
    70.00000000     82.79421637       0.01905000   5656.65829295 # SN1993ae
% wc snia.data
     595    3570   45340 snia.data
```

Try following practice.

Practice 15-19

Make your own Python script to find type-Ia supernovae and print distance and velocity of those type-Ia supernovae.

7.7 Making a Hubble diagram from type-Ia supernovae

Make a Hubble diagram from type-Ia supernova data of Open Supernova Catalog.

Python Code 23: ai202209_s15_05_04.py

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

#
# Time-stamp: <2022/12/25 23:50:02 (CST) daisuke>
#

# importing argparse module
import argparse

# importing numpy module
import numpy

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

# command-line argument analysis
desc = 'making Hubble diagram using SNIa from Open Supernova Catalog'
parser = argparse.ArgumentParser (description=desc)
parser.add_argument ('-i', '--input', help='input data file name')
parser.add_argument ('-o', '--output', help='output figure file name')
```

```
args = parser.parse_args ()

# file names
file_data = args.input
file_fig  = args.output

# numpy arrays to store data
data_d = numpy.array ([])
data_v = numpy.array ([])
data_z = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # reading data
    for line in fh:
        # skip the line, if it starts with '#'
        if (line[0] == '#'):
            continue
        # splitting the line
        data  = line.split ()
        # extracting data
        d1 = float (data[0])
        d2 = float (data[1])
        z  = float (data[2])
        v  = float (data[3])
        # checking distance
        diff = numpy.absolute (d1 - d2)
        diff_rel = diff / d2
        if (diff_rel > 0.1):
            continue
        # appending data to numpy arrays
        data_d = numpy.append (data_d, d2)
        data_v = numpy.append (data_v, v)
        data_z = numpy.append (data_z, z)

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

# axes
ax.set_xlabel ('Distance [Mpc]')
ax.set_ylabel ('Velocity [km/s]')
ax.grid ()

# making a Hubble diagram
ax.plot (data_d, data_v, \
         linestyle='None', marker='o', color='blue', markersize=2, \
         label='type-Ia supernovae')
ax.legend ()

# saving the plot into a file
fig.savefig (file_fig, dpi=225, bbox_inches="tight")
```

Execute above script to make a Hubble diagram.

```
% chmod a+x ai202209_s15_05_04.py
% ./ai202209_s15_05_04.py -i snia.data -o snia_hd.png
% ls -lF snia_hd.png
```

```
-rw-r--r--  1 daisuke  taiwan  73116 Dec 26 00:14 snia_hd.png
```

Display the PNG file. (Fig. 17)

```
% feh -dF snia_hd.png
```

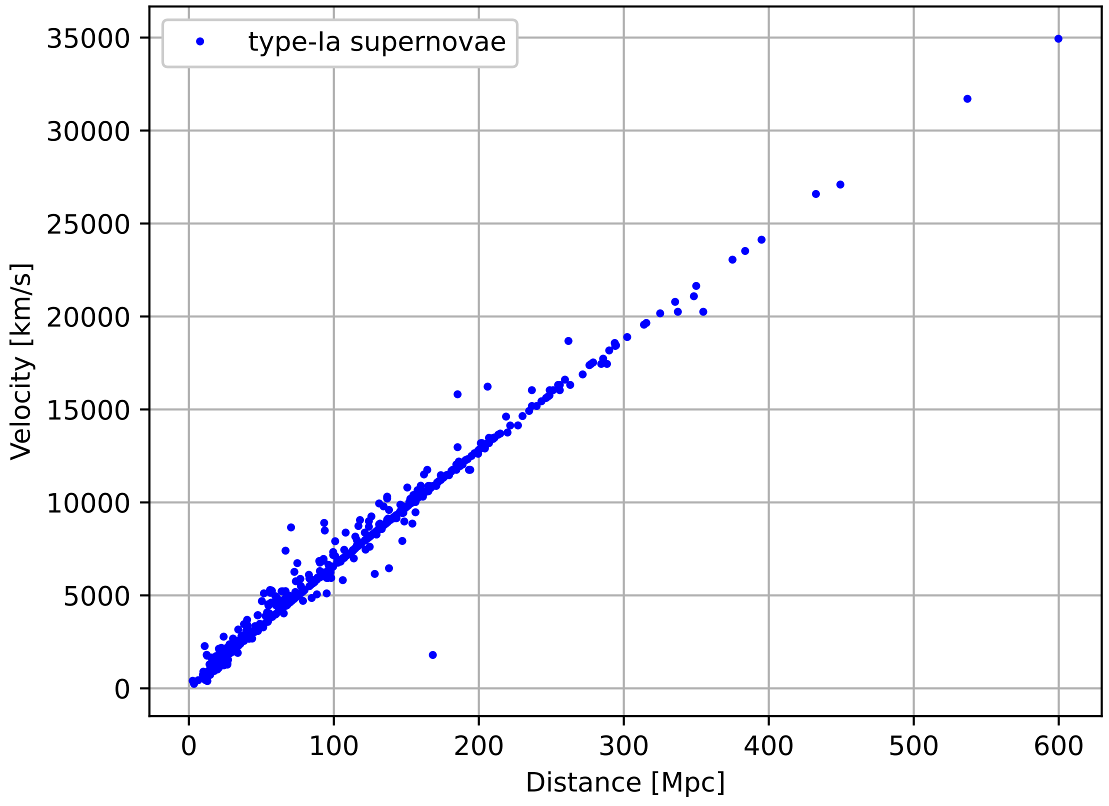


Figure 17: The Hubble diagram constructed from Open Supernova Catalog.

Try following practice.

Practice 15-20

Change marker shape, size, and colour and make the plot again.

7.8 Determination of Hubble constant

Make a Python script to carry out least-squares method to determine the value of Hubble constant.

Python Code 24: ai202209_s15_05_05.py

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

#
# Time-stamp: <2022/12/26 00:18:23 (CST) daisuke>
#

# importing argparse module
import argparse
```

```
# importing numpy module
import numpy

# importing scipy module
import scipy.optimize
import scipy.constants

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

# command-line argument analysis
desc = 'determination of Hubble constant from type-Ia SN data'
parser = argparse.ArgumentParser (description=desc)
parser.add_argument ('-i', '--input', help='input data file name')
parser.add_argument ('-o', '--output', help='output figure file name')
args = parser.parse_args ()

# file names
file_data = args.input
file_fig  = args.output

# numpy arrays to store data
data_d = numpy.array ([])
data_v = numpy.array ([])
data_z = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # reading data
    for line in fh:
        # skip the line, if it starts with '#'
        if (line[0] == '#'):
            continue
        # splitting the line
        data  = line.split ()
        # extracting data
        d1 = float (data[0])
        d2 = float (data[1])
        z  = float (data[2])
        v  = float (data[3])
        # checking distance
        diff = numpy.absolute (d1 - d2)
        diff_rel = diff / d2
        if (diff_rel > 0.1):
            continue
        # appending data to numpy arrays
        data_d = numpy.append (data_d, d2)
        data_v = numpy.append (data_v, v)
        data_z = numpy.append (data_z, z)

# initial values of coefficient
H0 = 100.0
init = [H0]

# function
def func (x, H0):
    y = H0 * x
    return (y)
```

```
# least-squares method
popt, pcov = scipy.optimize.curve_fit (func, data_d, data_v, p0=init)

# result of fitting
H0_bestfit = popt[0]
print ("popt:")
print (popt)
print ("pcov:")
print (pcov)

# degree of freedom
dof = len (data_d) - len (init)
print (f"dof = {dof}")

# residual
residual = data_v - func (data_d, popt[0])
reduced_chi2 = (residual**2).sum () / dof
print (f"reduced chi^2 = {reduced_chi2}")

# error of H0
H0_err = numpy.sqrt (pcov[0][0])
print (f"H0 = {H0_bestfit} +/- {H0_err} ({H0_err / H0_bestfit * 100.0} %)")

# fitted curve
fitted_x = numpy.linspace (0.0, 700.0, 1000)
fitted_y = func (fitted_x, H0_bestfit)

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

# axes
ax.set_xlabel ('Distance [Mpc]')
ax.set_ylabel ('Velocity [km/s]')
ax.grid ()

# making a Hubble diagram
label_fitting = f"best-fit line (H0={H0_bestfit:4.1f})"
ax.plot (data_d, data_v, \
         linestyle='None', marker='o', color='blue', markersize=2, \
         label='type-Ia supernovae')
ax.plot (fitted_x, fitted_y, linestyle='--', color='red', \
         label=label_fitting)
ax.legend ()

# saving the plot into a file
fig.savefig (file_fig, dpi=225, bbox_inches="tight")
```

Execute above script to determine the Hubble constant using least-squares method.

```
% chmod a+x ai202209_s15_05_05.py
% ./ai202209_s15_05_05.py -i snia.data -o snia_fit.png
popt:
[64.06753145]
pcov:
[[0.05332936]]
dof = 550
```

```
reduced chi^2 = 573105.7427116291
H0 = 64.06753144502201 +/- 0.23093151460240094 (0.3604501521969348 %)
```

The best-fit Hubble constant is 64.07 ± 0.23 km s^{-1} Mpc^{-1}.
Display the PNG file. (Fig. 18)

```
% feh -dF snia_fit.png
```

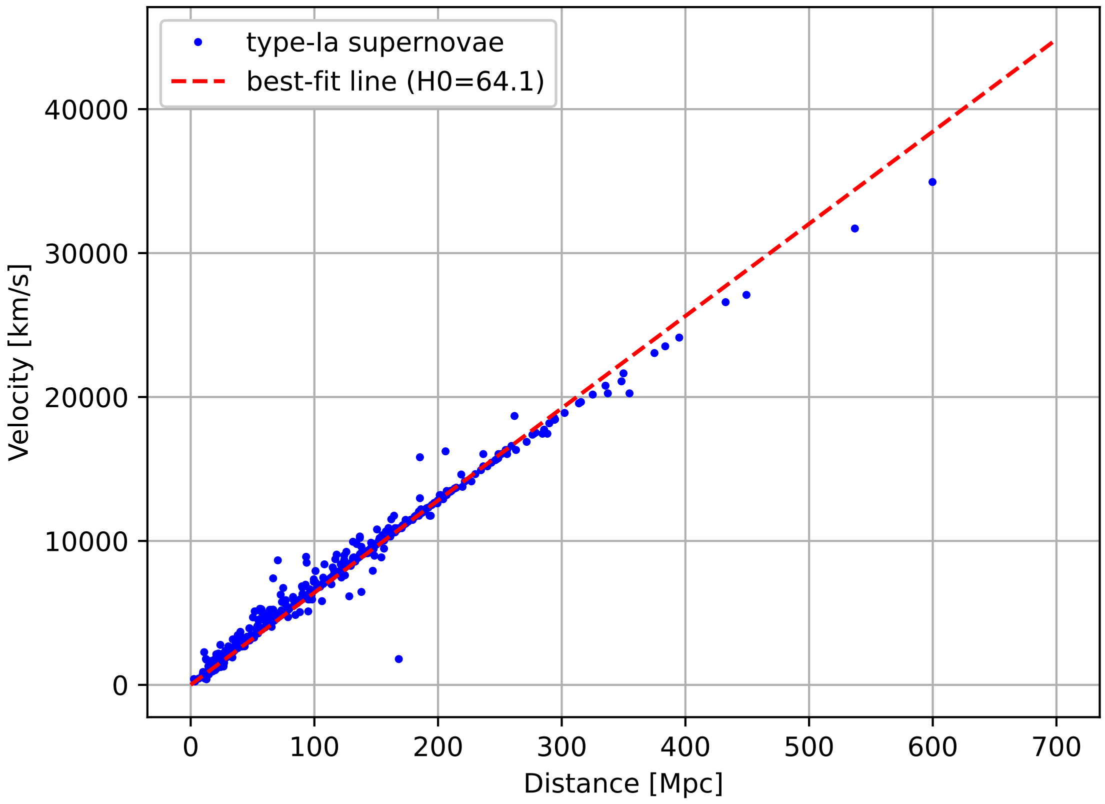


Figure 18: The Hubble diagram constructed from Open Supernova Catalog. The best-fit Hubble constant is determined to be 64.07 ± 0.23 km s^{-1} Mpc^{-1}.

Try following practice.

Practice 15-21

Change the initial value for the fitting coefficient and carry out least-squares method again.

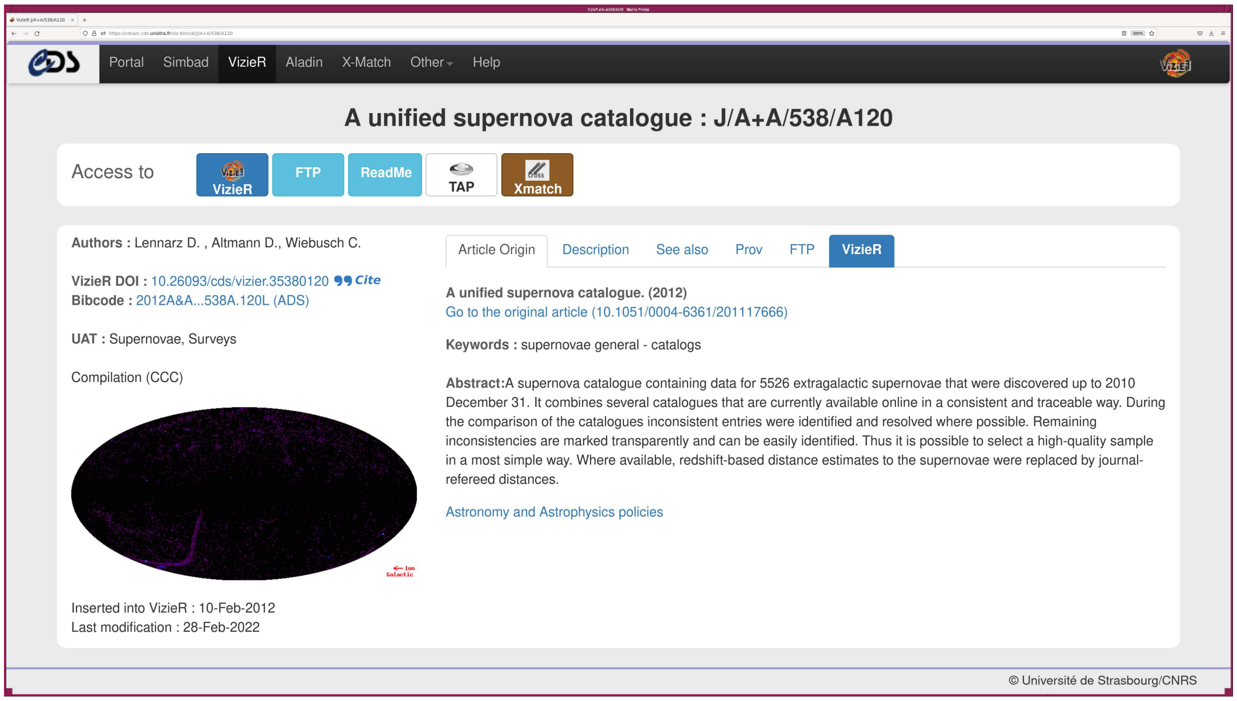
8 Unified Supernovae Catalogue

The Unified Supernovae Catalogue can be found at following address. (Fig. 19)

- Unified Supernovae Catalogue: `https://cdsarc.cds.unistra.fr/viz-bin/cat/J/A+A/538/A120`

8.1 Downloading the catalogue data

Make a Python script to download the catalogue data.



A unified supernova catalogue : J/A+A/538/A120

Authors : Lennarz D. , Altmann D., Wiebusch C.

VizieR DOI : 10.26093/cds/vizier.35380120 Cite
Bibcode : 2012A&A...538A.120L (ADS)

UAT : Supernovae, Surveys

Compilation (CCC)

Inserted into VizieR : 10-Feb-2012
Last modification : 28-Feb-2022

A unified supernova catalogue. (2012)
Go to the original article (10.1051/0004-6361/201117666)

Keywords : supernovae general - catalogs

Abstract:A supernova catalogue containing data for 5526 extragalactic supernovae that were discovered up to 2010 December 31. It combines several catalogues that are currently available online in a consistent and traceable way. During the comparison of the catalogues inconsistent entries were identified and resolved where possible. Remaining inconsistencies are marked transparently and can be easily identified. Thus it is possible to select a high-quality sample in a most simple way. Where available, redshift-based distance estimates to the supernovae were replaced by journal-refereed distances.

Astronomy and Astrophysics policies

© Université de Strasbourg/CNRS

Figure 19: The CDS web page for downloading Unified Supernova Catalogue.

Python Code 25: ai202209_s15_06_00.py

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

#
# Time-stamp: <2022/12/26 00:26:10 (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/J/A+A/538/A120/usc.dat'

# output file name
file_output = 'usc.data'

# 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 data.

```
% chmod a+x ai202209_s15_06_00.py
% ./ai202209_s15_06_00.py
Now, fetching https://cdsarc.cds.unistra.fr/ftp/J/A+A/538/A120/usc.dat...
Finished fetching https://cdsarc.cds.unistra.fr/ftp/J/A+A/538/A120/usc.dat!
Now, writing the data into file "usc.data"...
Finished writing the data into file "usc.data"!
% ls -lF usc.data
-rw-r--r--  1 daisuke  taiwan  1459482 Dec 26 00:26 usc.data
% head usc.data | cut -b 1-80
2010ma 1                                  00 48 55.35 -34 33 59.5
2010lz                                    01 50 21.32 -21 44 31.9
2010ly                                    03 05 03.56 -19 40 38.8
2010lx                                    04 44 44.57 -22 12 19.1    0.4E    1.4
2010lw   (SDSSJ131121.61-12141.5)         13 11 21.57 -01 21 41.2    0.77W   0.3
2010lv                                    13 19 42.89 -33 15 18.5    1.2E    4.9
2010lu   (SDSSJ090600.26+292041.0)        09 06 00.70 +29 20 32.5    5.8E    8.3
2010lt   UGC3378                          06 06 09.18 +83 50 28.8    20.W    10.
2010ls   (NGC5266A)                       13 40 51.56 -48 23 03.1
2010lr   (PGC132726)                      00 02 35.50 -30 43 52.4    19.4E   13.
```

8.2 Finding type-Ia supernovae

Make a Python script to read the data file of Unified Supernovae Catalogue and find type-Ia supernovae. Here is an example.

Python Code 26: ai202209_s15_06_01.py

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

#
# Time-stamp: <2022/12/26 00:30:03 (CST) daisuke>
#

# importing scipy module
import scipy.constants

# data file
file_data = 'usc.data'

# constant
c = scipy.constants.c

# opening file
with open (file_data, 'r') as fh:
    # reading file
    for line in fh:
        # extracting data
        name_str     = line[0:6].strip ()
        type_str     = line[86:104].strip ()
        z_str        = line[226:234].strip ()
        dist_str     = line[237:244].strip ()
        dist_err_str = line[245:251].strip ()

        # skip if redshift is missing
        if (z_str == ''):
            continue
        # skip if distance is missing
        if (dist_str == ''):
            continue
        # skip if type is not Ia
        if not (type_str == 'Ia'):
            continue

        # conversion from string to float
        z = float (z_str)
        dist = float (dist_str)
        dist_err = float (dist_err_str)
        dist_err_rel = dist_err / dist

        # calculation of velocity from redshift
        v = ( (z + 1.0)**2 - 1.0 ) / ( (z + 1.0)**2 + 1.0 ) * c * 10**-3

        # skip if redshift is negative
        if (z < 0.0):
            continue

        # skip if distance measurement is not accurate
        if (dist_err_rel > 0.1):
            continue

        # printing data
        print (f"{dist:15.8f} {dist_err:15.8f} {z:15.8f} {v:15.8f} # {name_str}")
```

Execute above script to read the data file and find type-Ia supernovae.

```
% chmod a+x ai202209_s15_06_01.py
% ./ai202209_s15_06_01.py > usc_snia.data
% ls -lF usc_snia.data
-rw-r--r--  1 daisuke  taiwan  17009 Dec 26 00:32 usc_snia.data
% head usc_snia.data
    25.00000000      1.00000000      0.00720000   2150.73527706 # 2010ih
    33.00000000      2.00000000      0.00860000   2567.12922016 # 2010bv
    18.00000000      1.00000000      0.00390000   1166.91068183 # 2008ge
    38.40000000      0.40000000      0.00940000   2804.80485409 # 2008fv
    66.00000000      3.00000000      0.01600000   4758.31072722 # 2008eo
   101.00000000      9.00000000      0.02500000   7401.15486071 # 2008bf
   146.00000000     12.00000000      0.03300000   9730.00011841 # 2008af
    28.00000000      2.00000000      0.00810000   2418.48453824 # 2008Q
    16.00000000      1.00000000      0.00550000   1644.32422628 # 2007sr
    18.10000000      0.60000000      0.00650000   1942.31799425 # 2007on
```

Try following practice.

Practice 15-22

Make your own Python script to find type-Ia supernovae.

8.3 Making a Hubble diagram

Make a Python script to make a Hubble diagram.

Python Code 27: ai202209_s15_06_02.py

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

#
# Time-stamp: <2022/12/26 00:34:02 (CST) daisuke>
#

# importing argparse module
import argparse

# importing numpy module
import numpy

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

# command-line argument analysis
desc = 'making Hubble diagram using Unified Supernova Catalogue'
parser = argparse.ArgumentParser (description=desc)
parser.add_argument ('-i', '--input', help='input data file name')
parser.add_argument ('-o', '--output', help='output figure file name')
args = parser.parse_args ()

# file names
file_data = args.input
file_fig  = args.output

# numpy arrays to store data
data_d     = numpy.array ([])
data_d_err = numpy.array ([])
data_z     = numpy.array ([])
data_v     = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # reading data
    for line in fh:
        # skip the line, if it starts with '#'
        if (line[0] == '#'):
            continue
        # splitting the line
        data  = line.split ()
        # extracting data
        d     = float (data[0])
        d_err = float (data[1])
        z     = float (data[2])
        v     = float (data[3])
        # appending data to numpy arrays
        data_d     = numpy.append (data_d, d)
        data_d_err = numpy.append (data_d_err, d)
        data_z      = numpy.append (data_z, z)
        data_v      = numpy.append (data_v, v)

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

# axes
ax.set_xlabel ('Distance [Mpc]')
ax.set_ylabel ('Velocity [km/s]')
ax.grid ()

# making a Hubble diagram
ax.plot (data_d, data_v, \
         linestyle='None', marker='o', color='blue', markersize=2, \
         label='type-Ia supernovae from USC')
ax.legend ()

# saving the plot into a file
fig.savefig (file_fig, dpi=225, bbox_inches="tight")
```

Execute above script to make a Hubble diagram.

```
% chmod a+x ai202209_s15_06_02.py
% ./ai202209_s15_06_02.py -i usc_snia.data -o usc_hd.png
% ls -lF usc_hd.png
-rw-r--r--  1 daisuke  taiwan  72125 Dec 26 00:34 usc_hd.png
```

Display the PNG file. (Fig. 20)

```
% feh -dF usc_hd.png
```

Try following practice.

Practice 15-23

Change marker shape, size, and colour and make the plot again.

8.4 Determination of Hubble constant

Make a Python script to determine Hubble constant.

Python Code 28: ai202209_s15_06_03.py

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

#
# Time-stamp: <2022/12/26 00:43:19 (CST) daisuke>
#

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing scipy module
```

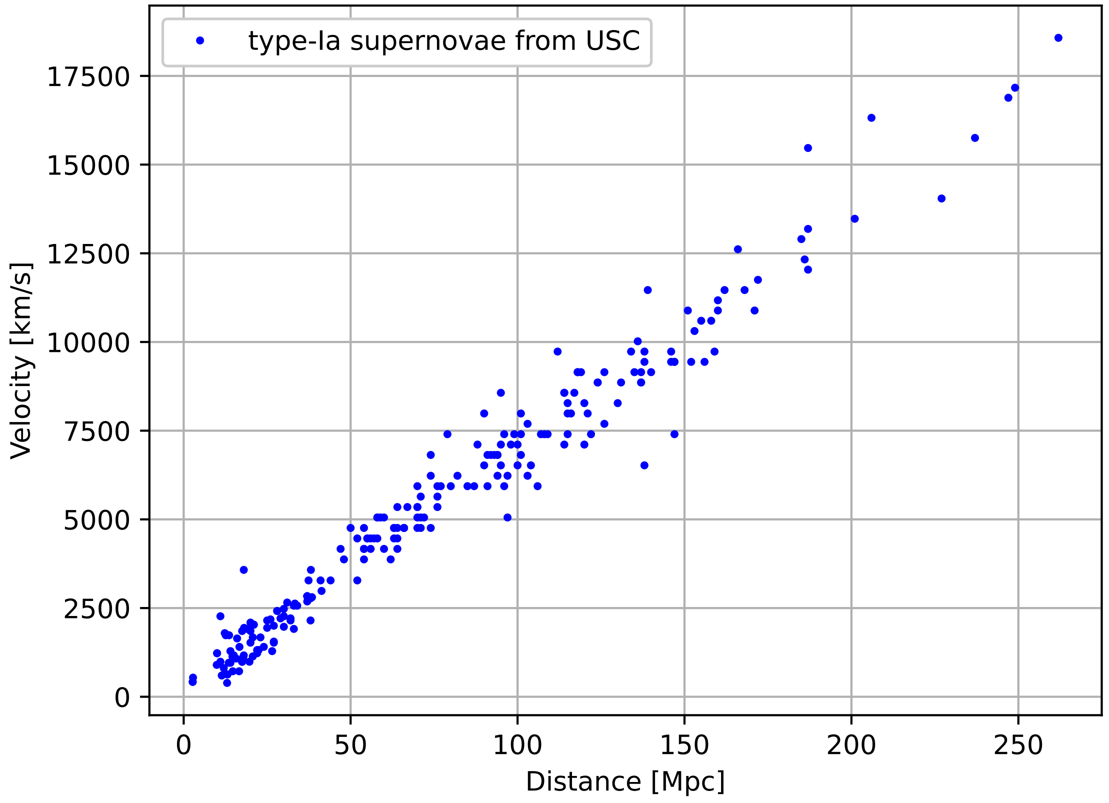


Figure 20: The Hubble diagram constructed from Unified Supernova Catalogue.

```
import scipy.optimize
import scipy.constants

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

# command-line argument analysis
parser = argparse.ArgumentParser \
    (description='Hubble constant from Unified Supernova Catalogue')
parser.add_argument ('-i', '--input', help='input data file name')
parser.add_argument ('-o', '--output', help='output figure file name')
args = parser.parse_args ()

# file names
file_data = args.input
file_fig  = args.output

# numpy arrays to store data
data_d     = numpy.array ([])
data_d_err = numpy.array ([])
data_z     = numpy.array ([])
data_v     = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # reading data
    for line in fh:
        # skip the line, if it starts with '#'
        if (line[0] == '#'):
            continue
        # splitting the line
        data  = line.split ()
        # extracting data
        d     = float (data[0])
        d_err = float (data[1])
        z     = float (data[2])
        v     = float (data[3])
        # appending data to numpy arrays
        data_d     = numpy.append (data_d, d)
        data_d_err = numpy.append (data_d_err, d)
        data_z     = numpy.append (data_z, z)
        data_v     = numpy.append (data_v, v)

# initial values of coefficient
H0 = 100.0
init = [H0]

# function
def func (x, H0):
    y = H0 * x
    return (y)

# least-squares method
popt, pcov = scipy.optimize.curve_fit (func, data_d, data_v, p0=init)

# result of fitting
H0_bestfit = popt[0]
print ("popt:")
print (popt)
print ("pcov:")
print (pcov)

# degree of freedom
dof = len (data_d) - len (init)
print (f"dof = {dof}")

# residual
residual = data_v - func (data_d, popt[0])
reduced_chi2 = (residual**2).sum () / dof
print (f"reduced chi^2 = {reduced_chi2}")

# error of H0
H0_err = numpy.sqrt (pcov[0][0])
print (f"H0 = {H0_bestfit} +/- {H0_err} ({H0_err / H0_bestfit * 100.0} %)")

# fitted curve
fitted_x = numpy.linspace (0.0, 300.0, 1000)
fitted_y = func (fitted_x, H0_bestfit)

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

# axes
ax.set_xlabel ('Distance [Mpc]')
ax.set_ylabel ('Velocity [km/s]')
ax.grid ()

# making a Hubble diagram
label_fitting = "best-fit line (H0=%4.1f)" % (H0_bestfit)
ax.plot (data_d, data_v, \
         linestyle='None', marker='o', color='blue', markersize=2, \
         label='type-Ia supernovae from USC')
ax.plot (fitted_x, fitted_y, linestyle='--', color='red', \
         label=label_fitting)
ax.legend ()

# saving the plot into a file
fig.savefig (file_fig, dpi=225, bbox_inches="tight")
```

Execute above script to find the best-fit Hubble constant using least-squares method.

```
% chmod a+x ai202209_s15_06_03.py
% ./ai202209_s15_06_03.py -i usc_snia.data -o usc_fit.png
% ls -lF usc_fit.png
```

The best-fit Hubble constant is determined to be 69.65 $\pm$ 0.51 km s^{-1} Mpc^{-1}.
Display the PNG file. (Fig. 21)

```
% feh -dF usc_fit.png
```

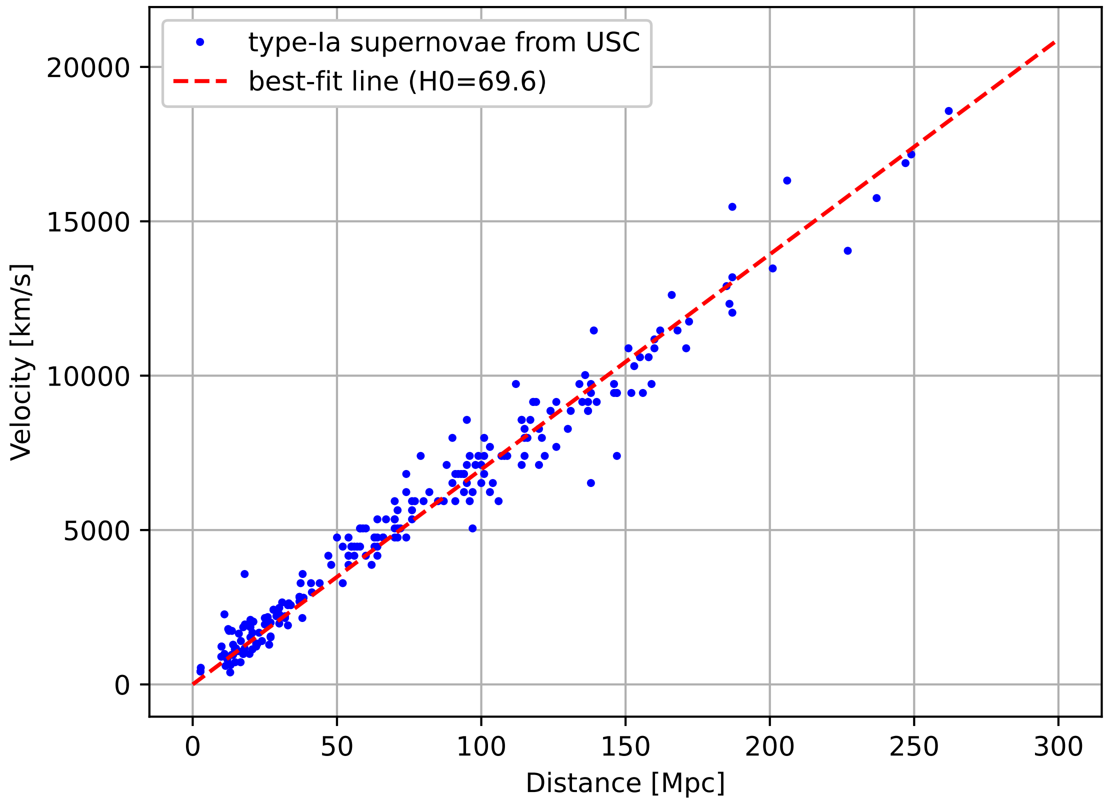


Figure 21: The Hubble diagram constructed from Unified Supernova Catalogue and derived best-fit Hubble constant.

Try following practice.

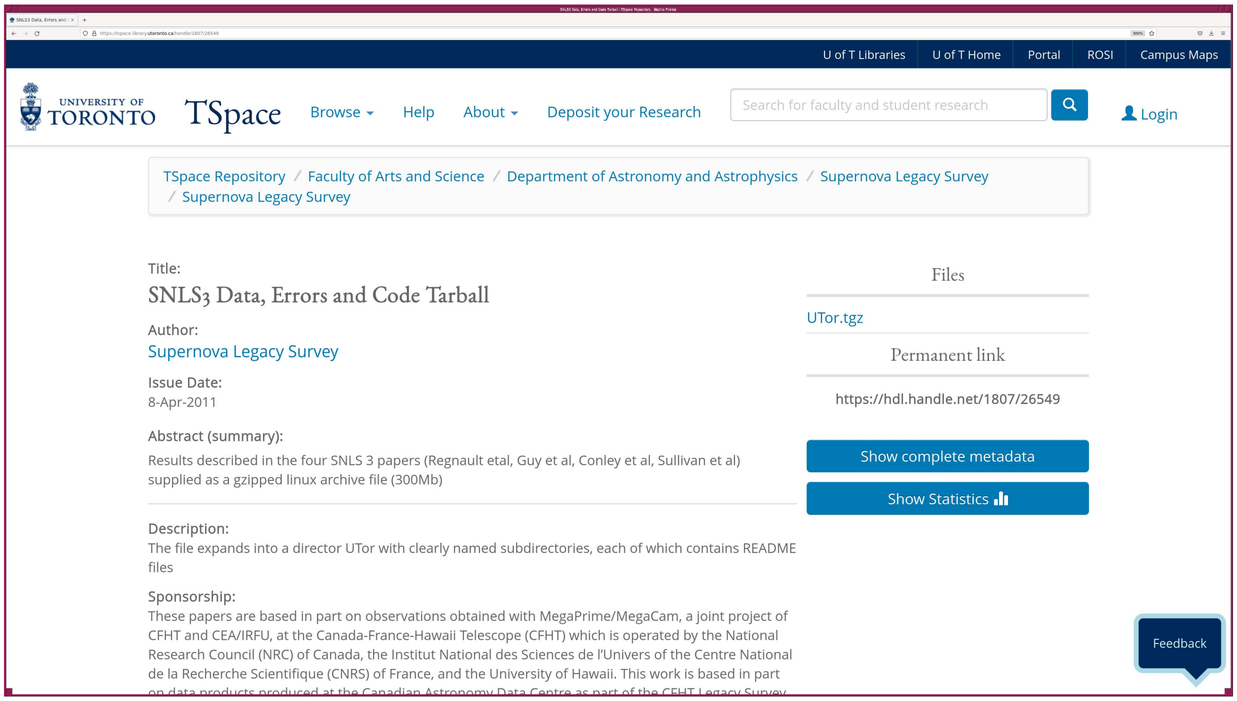
Practice 15-24

Change the initial value for the fitting coefficient and carry out least-squares method again.

9 Cosmological parameters

The data of high-z supernovae data of Supernova Legacy Survey can be found at following address. (Fig. 22)

- Supernova Legacy Survey: `http://hdl.handle.net/1807/26549`



TSpace Repository / Faculty of Arts and Science / Department of Astronomy and Astrophysics / Supernova Legacy Survey / Supernova Legacy Survey

Title:
SNLS3 Data, Errors and Code Tarball

Author:
Supernova Legacy Survey

Issue Date:
8-Apr-2011

Abstract (summary):
Results described in the four SNLS 3 papers (Regnault etal, Guy et al, Conley et al, Sullivan et al) supplied as a gzipped linux archive file (300Mb)

Description:
The file expands into a director UTor with clearly named subdirectories, each of which contains README files

Sponsorship:
These papers are based in part on observations obtained with MegaPrime/MegaCam, a joint project of CFHT and CEA/IRFU, at the Canada-France-Hawaii Telescope (CFHT) which is operated by the National Research Council (NRC) of Canada, the Institut National des Sciences de l'Univers of the Centre National de la Recherche Scientifique (CNRS) of France, and the University of Hawaii. This work is based in part on data products produced at the Canadian Astronomy Data Centre as part of the CFHT Legacy Survey

UTor.tgz

https://hdl.handle.net/1807/26549

Figure 22: Supernova Legacy Survey data.

9.1 Downloading high-z supernovae data

Make a Python script to download high-z supernovae data of Supernova Legacy Survey. Here is an example.

Python Code 29: ai202209_s15_07_00.py

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

#
# Time-stamp: <2022/12/26 00:53:15 (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://tspace.library.utoronto.ca/bitstream/1807/26549/1/UTor.tgz'

# output file name
file_output = 'UTor.tgz'

# 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 data.

```
% chmod a+x ai202209_s15_07_00.py
% ./ai202209_s15_07_00.py
Now, fetching https://tspace.library.utoronto.ca/bitstream/1807/26549/1/UTor.tgz
...
Finished fetching https://tspace.library.utoronto.ca/bitstream/1807/26549/1/UTor
.tgz!
Now, writing the data into file "UTor.tgz"...
Finished writing the data into file "UTor.tgz"!
% ls -lF UTor.tgz
-rw-r--r--  1 daisuke  taiwan  317425529 Dec 26 00:59 UTor.tgz
```

9.2 Extracting data

Extract data.

```
% tar xzvf UTor.tgz
% ls -F UTor
README              covariances/        lightcurve_params/  results/
code/               figures/            photometry/         spectra/
% ls -F UTor/lightcurve_params/
README
combined.dat
salt2.dat
sifto.dat
snls_malmcorr.txt
snlsc_0_20k_0.5_psczsc_120,-104,120.clusvp1
% head UTor/lightcurve_params/combined.dat | cut -b 1-80
#name        zcmb     zhel     dz       mb         dmb      s        ds       col
sn2004s      0.01029  0.00937  0.00002  14.18292  0.04230  0.97304  0.02643   0.
```

```
sn1999ac     0.01006  0.00950  0.00002  14.12950  0.03029  0.98663  0.00915   0.
sn1997do     0.01055  0.01012  0.00002  14.31719  0.03558  0.98284  0.02342   0.
sn2006bh     0.01113  0.01090  0.00100  14.34661  0.02136  0.81423  0.00826  -0.
sn2002dp     0.01090  0.01164  0.00002  14.59660  0.02968  0.97303  0.02921   0.
sn2005al     0.01231  0.01240  0.00100  14.84347  0.02586  0.87122  0.01307  -0.
sn2001ep     0.01334  0.01301  0.00000  14.90363  0.02510  0.90252  0.02379   0.
sn1997e      0.01426  0.01354  0.00012  15.11765  0.04133  0.81887  0.01701   0.
sn2001fe     0.01457  0.01354  0.00002  14.68486  0.02404  1.07722  0.02806  -0.
```

9.3 Making a Hubble diagram

Make a Python script to make a Hubble diagram using SNLS data. Here is an example.

Python Code 30: ai202209_s15_07_01.py

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

#
# Time-stamp: <2022/12/26 01:08:11 (CST) daisuke>
#

# importing argparse module
import argparse

# importing numpy module
import numpy

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

# command-line argument analysis
desc = 'Hubble diagram of high-z supernovae'
parser = argparse.ArgumentParser (description=desc)
parser.add_argument ('-i', '--input', help='input data file name')
parser.add_argument ('-o', '--output', help='output figure file name')
args = parser.parse_args ()

# file names
file_data = args.input
file_fig  = args.output

# numpy arrays for storing data
data_zcmb   = numpy.array ([])
data_mB     = numpy.array ([])
data_mB_err = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # reading file
    for line in fh:
        # skip if the line starts with '#'
        if (line[0] == '#'):
            continue
        # splitting the line
        data = line.split ()
        # extracting data
        zcmb   = float (data[1])
        mB     = float (data[4])
        mB_err = float (data[5])
        # appending data to numpy arrays
        data_zcmb   = numpy.append (data_zcmb, zcmb)
        data_mB     = numpy.append (data_mB, mB)
        data_mB_err = numpy.append (data_mB_err, mB_err)

# distance modulus
data_mu = data_mB + 19.30

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

# axes
ax.set_xlabel ('Redshift')
ax.set_ylabel ('Distance Modulus [mag]')
ax.grid ()

# making a Hubble diagram
ax.errorbar (data_zcmb, data_mu, yerr=data_mB_err, \
             marker='o', color='blue', markersize=1, linestyle='none', \
             ecolor='black', capsize=2, \
             label='high-z supernovae from SNLS')
ax.legend ()

# saving the plot into a file
fig.savefig (file_fig, dpi=225, bbox_inches="tight")
```

Execute above script to make a Hubble diagram.

```
% chmod a+x ai202209_s15_07_01.py
% ./ai202209_s15_07_01.py -i UTor/lightcurve_params/combined.dat -o snls_hd.png
% ls -lF snls_hd.png
-rw-r--r--  1 daisuke  taiwan  85085 Dec 26 01:08 snls_hd.png
```

Display the PNG file. (Fig. 23)

```
% feh -dF snls_hd.png
```

Try following practice.

Practice 15-25

Change marker shape, size, and colour and make the plot again.

9.4 Comparison with cosmology models

Compare high-z supernovae data with cosmology models. One model is a flat universe with 70% dark energy and 30% matter, and the other model is a flat universe with 100% matter.

Python Code 31: ai202209_s15_07_02.py

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

#
# Time-stamp: <2022/12/26 01:19:47 (CST) daisuke>
#
```

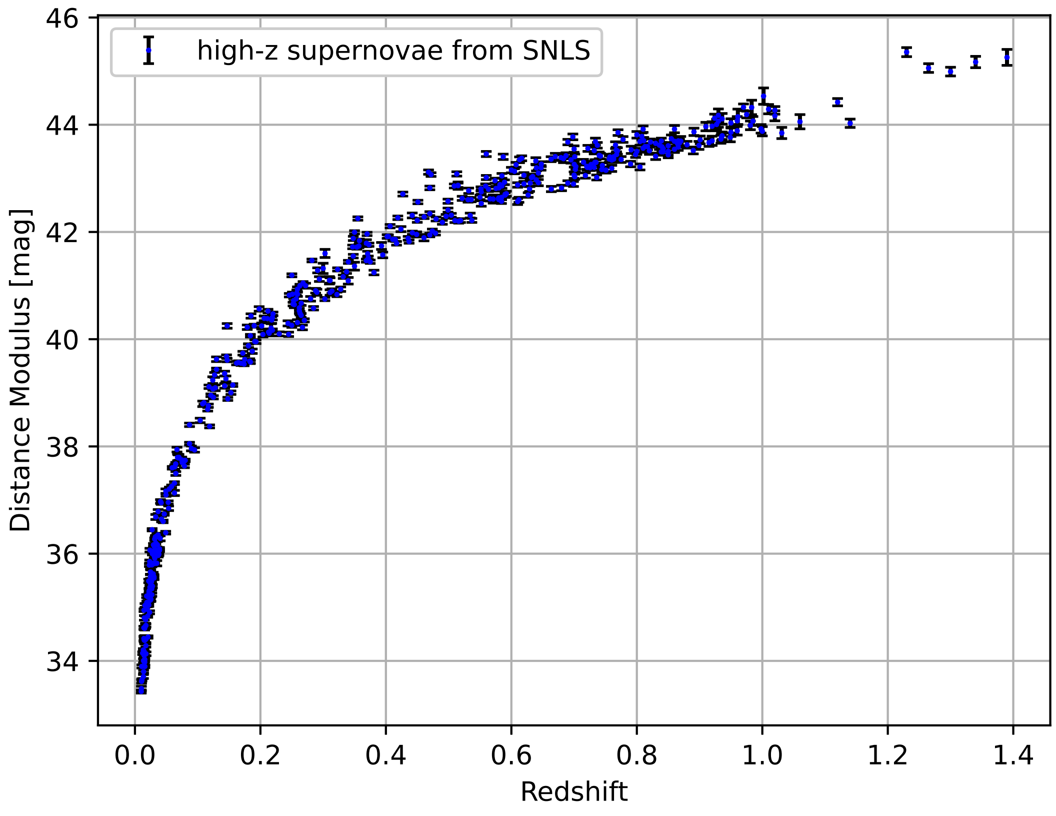


Figure 23: The Hubble diagram constructed from Supernova Legacy Survey.

```
# importing argparse module
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy.cosmology

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

# command-line argument analysis
desc = 'Hubble diagram of high-z supernovae'
parser = argparse.ArgumentParser (description=desc)
parser.add_argument ('-i', '--input', help='input data file name')
parser.add_argument ('-o', '--output', help='output figure file name')
args = parser.parse_args ()

# file names
file_data = args.input
file_fig  = args.output

# numpy arrays for storing data
data_zcmb    = numpy.array ([])
data_mB      = numpy.array ([])
```

```
data_mB_err = numpy.array ([])

# opening file
with open (file_data, 'r') as fh:
    # reading file
    for line in fh:
        # skip if the line starts with '#'
        if (line[0] == '#'):
            continue
        # splitting the line
        data = line.split ()
        # extracting data
        zcmb   = float (data[1])
        mB     = float (data[4])
        mB_err = float (data[5])
        # appending data to numpy arrays
        data_zcmb   = numpy.append (data_zcmb, zcmb)
        data_mB     = numpy.append (data_mB, mB)
        data_mB_err = numpy.append (data_mB_err, mB_err)

# distance modulus
data_mu = data_mB + 19.30

# cosmology models
cosmo_000_100 = astropy.cosmology.FlatLambdaCDM (H0=70.0, Om0=1.0)
cosmo_070_030 = astropy.cosmology.FlatLambdaCDM (H0=70.0, Om0=0.3)

# redshift
z_min = -2.0
z_max = 1.0
z = numpy.logspace (z_min, z_max, num=300)

# distance modulus
distmod_000_100 = cosmo_000_100.distmod (z)
distmod_070_030 = cosmo_070_030.distmod (z)

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

# axes
ax.set_xlabel ('Redshift')
ax.set_ylabel ('Distance Modulus [mag]')
ax.grid ()
ax.set_xlim (0.0, 1.5)
ax.set_ylim (32.5, 47.0)

# making a Hubble diagram
ax.errorbar (data_zcmb, data_mu, yerr=data_mB_err, \
             linestyle='None', marker='o', color='blue', markersize=1, \
             ecolor='black', capsize=2, \
             label='high-z supernovae from SNLS')
ax.plot (z, distmod_000_100, linestyle='-', linewidth=3, color='g', \
         label='$\Omega_\Lambda=0.0$, $\Omega_m=1.0$, flat Universe')
ax.plot (z, distmod_070_030, linestyle='--', linewidth=3, color='r', \
         label='$\Omega_\Lambda=0.7$, $\Omega_m=0.3$, flat Universe')
ax.legend (loc='lower right')
```

```
# saving the plot into a file
fig.savefig (file_fig, dpi=225, bbox_inches="tight")
```

Execute above script to show cosmology models.

```
% chmod a+x ai202209_s15_07_02.py
% ./ai202209_s15_07_02.py -i UTor/lightcurve_params/combined.dat -o snls_cosmo.png
% ls -lF snls_cosmo.png
-rw-r--r--  1 daisuke  taiwan  116094 Dec 26 01:19 snls_cosmo.png
```

Display the PNG file. (Fig. 24)

```
% feh -dF snls_cosmo.png
```

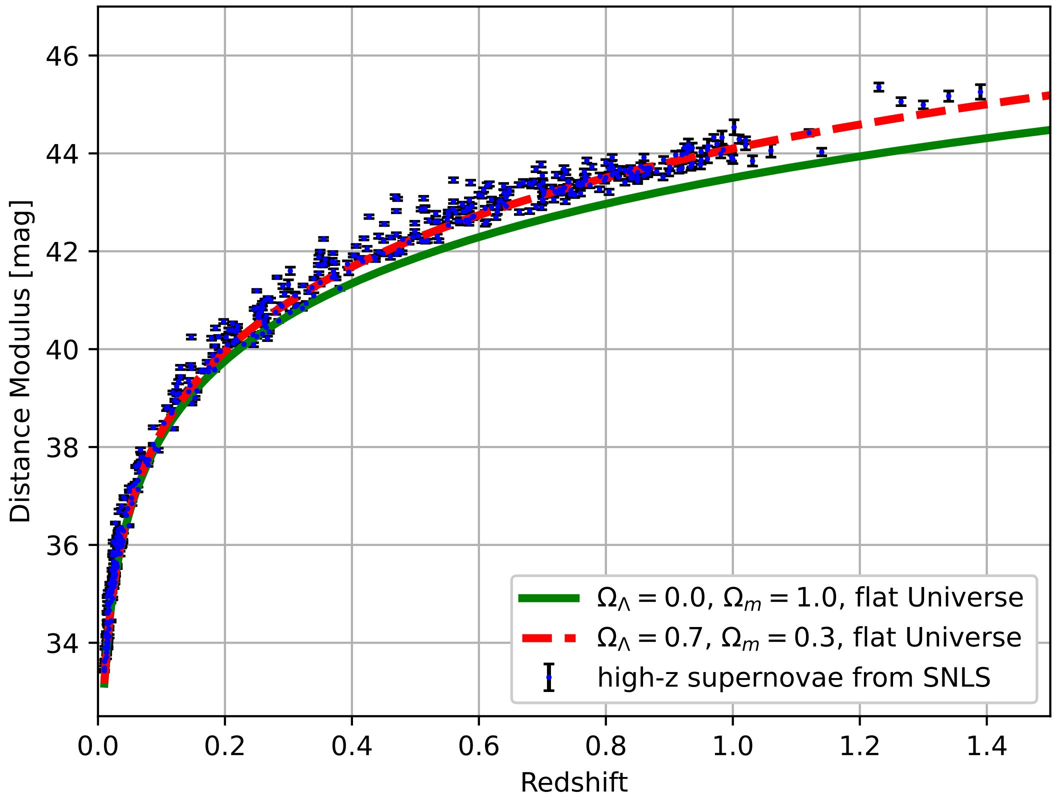


Figure 24: The Hubble diagram constructed from Supernova Legacy Survey and two cosmology models.

To learn more about "`astropy.cosmology`", visit following web page and read the document. (Fig. 25)

- `https://docs.astropy.org/en/stable/cosmology/`

Try following practice.

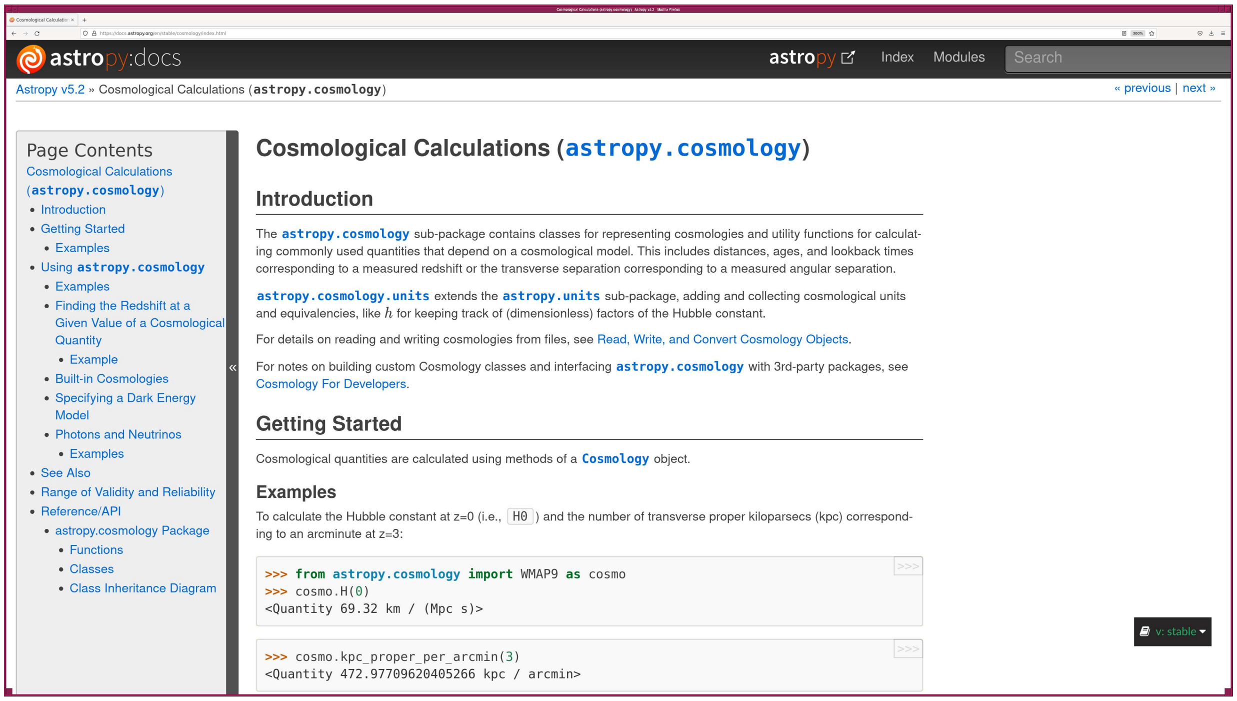
Practice 15-26

Make a cosmology model of 50% matter and 50% dark energy, and superimpose the model on the Hubble diagram.

10 Using astropy.cosmology module

The official documentation of `astropy.cosmology` module is available at following web page.

- `https://docs.astropy.org/en/stable/cosmology/`


astropy:docs

astropy Index Modules Search

Astropy v5.2 » Cosmological Calculations (`astropy.cosmology`) « previous | next »

Page Contents

- Cosmological Calculations (`astropy.cosmology`)
  - Introduction
  - Getting Started
    - Examples
  - Using `astropy.cosmology`
    - Examples
    - Finding the Redshift at a Given Value of a Cosmological Quantity
      - Example
    - Built-in Cosmologies
    - Specifying a Dark Energy Model
    - Photons and Neutrinos
      - Examples
  - See Also
  - Range of Validity and Reliability
  - Reference/API
    - astropy.cosmology Package
      - Functions
      - Classes
      - Class Inheritance Diagram

Cosmological Calculations (`astropy.cosmology`)

Introduction

The `astropy.cosmology` sub-package contains classes for representing cosmologies and utility functions for calculating commonly used quantities that depend on a cosmological model. This includes distances, ages, and lookback times corresponding to a measured redshift or the transverse separation corresponding to a measured angular separation.

`astropy.cosmology.units` extends the `astropy.units` sub-package, adding and collecting cosmological units and equivalencies, like h for keeping track of (dimensionless) factors of the Hubble constant.

For details on reading and writing cosmologies from files, see Read, Write, and Convert Cosmology Objects.

For notes on building custom Cosmology classes and interfacing `astropy.cosmology` with 3rd-party packages, see Cosmology For Developers.

Getting Started

Cosmological quantities are calculated using methods of a `Cosmology` object.

Examples

To calculate the Hubble constant at z=0 (i.e., `H0`) and the number of transverse proper kiloparsecs (kpc) corresponding to an arcminute at z=3:

```
>>> from astropy.cosmology import WMAP9 as cosmo
>>> cosmo.H(0)
<Quantity 69.32 km / (Mpc s)>
```

```
>>> cosmo.kpc_proper_per_arcmin(3)
<Quantity 472.97709620405266 kpc / arcmin>
```

v: stable

Figure 25: The official document for "`astropy.cosmology`" module.

10.1 Using a preloaded cosmology model

Try `astropy.cosmology` module. Show model parameters of Planck 2015 model.

Python Code 32: ai202209_s15_08_00.py

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

#
# Time-stamp: <2022/12/26 01:26:10 (CST) daisuke>
#

# import astropy module
import astropy.cosmology

# use preloaded cosmology model
cosmo_planck2015 = astropy.cosmology.Planck15

# Hubble constant and density parameters
H0   = cosmo_planck2015.H0
Ode0 = cosmo_planck2015.Ode0
Om0  = cosmo_planck2015.Om0
Odm0 = cosmo_planck2015.Odm0
Ob0  = cosmo_planck2015.Ob0
Og0  = cosmo_planck2015.Ogamma0
Onu0 = cosmo_planck2015.Onu0

Ot0  = Ode0 + Om0 + Og0 + Onu0

# printing parameters
print ("Parameters from Planck 2015")
print ("  H0   =", H0)
print ("  Ode0 =", Ode0)
print ("  Om0  =", Om0)
print ("  Odm0 =", Odm0)
print ("  Ob0  =", Ob0)
print ("  Og0  =", Og0)
print ("  Onu0 =", Onu0)
print ("  Ot0  =", Ot0)
```

Execute above script to show model parameters of Planck 2015.

```
% chmod a+x ai202209_s15_08_00.py
% ./ai202209_s15_08_00.py
Parameters from Planck 2015
  H0   = 67.74 km / (Mpc s)
  Ode0 = 0.6910098315260953
  Om0  = 0.3075
  Odm0 = 0.2589
  Ob0  = 0.0486
  Og0  = 5.38926326165147e-05
  Onu0 = 0.0014362758412881924
  Ot0  = 1.0
```

Try following practice.

Practice 15-27

Compare Planck 2015 model and high-z type-Ia supernovae measurements.

Make your own cosmology model. Here is an example.

Python Code 33: ai202209_s15_08_01.py

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

#
# Time-stamp: <2022/12/26 01:30:19 (CST) daisuke>
#

# import astropy module
import astropy.cosmology

# making your own cosmology model
cosmo = astropy.cosmology.FlatLambdaCDM (H0=70.0, Om0=0.30, Ob0=0.04)

# Hubble constant and density parameters
H0   = cosmo.H0
Ode0 = cosmo.Ode0
Om0  = cosmo.Om0
Odm0 = cosmo.Odm0
Ob0  = cosmo.Ob0
Og0  = cosmo.Ogamma0
Onu0 = cosmo.Onu0

Ot0  = Ode0 + Om0 + Og0 + Onu0

# Hubble time
hubble_time = cosmo.hubble_time

# printing parameters
print ("Parameters of your own cosmology model")
print ("  H0   =", H0)
print ("  Ode0 =", Ode0)
print ("  Om0  =", Om0)
print ("  Odm0 =", Odm0)
print ("  Ob0  =", Ob0)
print ("  Og0  =", Og0)
print ("  Onu0 =", Onu0)
print ("  Ot0  =", Ot0)
print ("  Hubble time =", hubble_time)
```

Execute above script.

```
% chmod a+x ai202209_s15_08_01.py
% ./ai202209_s15_08_01.py
Parameters of your own cosmology model
  H0   = 70.0 km / (Mpc s)
  Ode0 = 0.7
  Om0  = 0.3
  Odm0 = 0.26
  Ob0  = 0.04
  Og0  = 0.0
  Onu0 = 0.0
  Ot0  = 1.0
  Hubble time = 13.968460309725561 Gyr
```

Try following practice.

Practice 15-28

Compare your own cosmology model and high-z type-Ia supernovae measurements.

11 For your further reading

Read following document to learn more about JSON file I/O and Astropy.

- JSON encoder and decoder
  - `https://docs.python.org/3/library/json.html`
- Astropy: `https://docs.astropy.org/`
  - Data tables: `https://docs.astropy.org/en/stable/table/`
  - ASCII tables: `https://docs.astropy.org/en/stable/io/ascii/`
  - Cosmological Calculations: `https://docs.astropy.org/en/stable/cosmology/`

12 Assignment

1. The Big Bang
   (a) The Universe is believed to be originated from the Big Bang. List three observational evidences that support the Big Bang theory.
   (b) Give brief descriptions how those observational evidences support the Big Bang theory.
2. Hubble's law
   (a) What is Hubble flow?
   (b) What is Hubble's law?
   (c) What is Hubble constant? What is recently estimated value of Hubble constant? How the value was estimated?
   (d) What is Hubble time? What is recently estimated value of Hubble time? What is the age of the Universe?

   (e) What is Hubble distance? What is recently estimated value of Hubble distance?
   (f) What is peculiar velocity?

3. Methods of galaxy distance determination
   (a) What is a standard candle? What is cosmic distance ladder?
   (b) List five methods to determine distance of galaxies.
   (c) Explain each distance determination method.

4. type-Ia supernovae
   (a) What is type-Ia supernova? What is a progenitor object of a type-Ia supernova? How is a type-Ia supernova triggered? How different is a type-Ia supernova from other types of supernovae? How bright is it? How does typical lightcurve of type-Ia supernovae look like? How does typical spectrum of type-Ia supernovae look like?
   (b) How useful are type-Ia supernovae for cosmology?

5. What is distance modulus? What is the relationship between distance in Mpc and distance modulus?

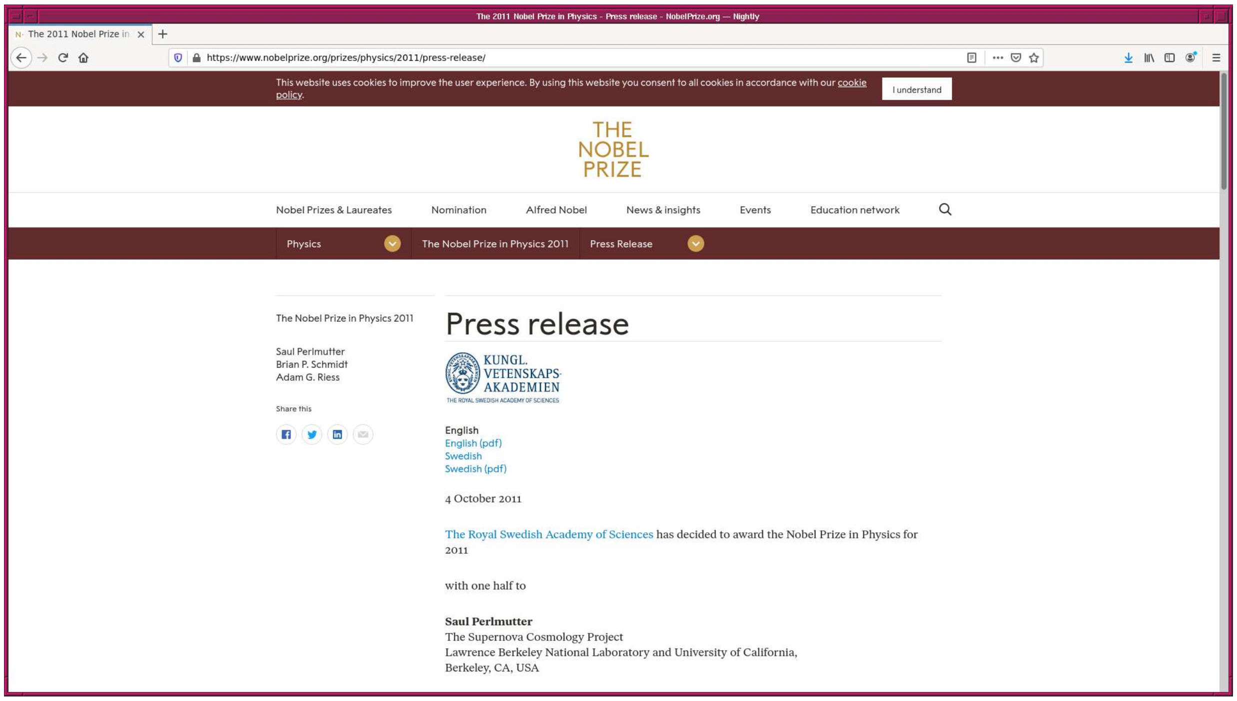
6. Doppler effect
   (a) What is Doppler effect?
   (b) Give three examples of astronomical Doppler shift observations and outputs from those measurements.
   (c) What is redshift z?
   (d) What is the relationship between redshift z, rest-frame and observed wavelengths of a spectral line?
   (e) What is the relationship between redshift z and recessional velocity v?

7. What is dark energy? What is cosmological constant?

8. Density parameters
   (a) What is density parameter?
   (b) What are recently estimated values of dark energy density parameter Ω_Λ, dark matter density parameter Ω_D, baryon density parameter Ω_B? How those values were estimated?

9. What is flat, open, and closed universe? What is critical density?

10. 2011 Nobel Prize in Physics was awarded to three astronomers for their research on accelerating expansion of the Universe. Explain what they did. What is accelerating expansion of the Universe? You may read followings (Fig. 26).
    - `https://www.nobelprize.org/uploads/2018/06/popular-physicsprize2011.pdf`
    - `https://www.nobelprize.org/uploads/2018/06/advanced-physicsprize2011.pdf`


The 2011 Nobel Prize in Physics - Press release - NobelPrize.org — Nightly
The 2011 Nobel Prize in
https://www.nobelprize.org/prizes/physics/2011/press-release/
I understand
THE NOBEL PRIZE
Nobel Prizes & Laureates
Nomination
Alfred Nobel
News & insights
Events
Education network
Physics
The Nobel Prize in Physics 2011
Press Release
Share this
KUNGL. VETENSKAPS-AKADEMIEN
THE ROYAL SWEDISH ACADEMY OF SCIENCES

This website uses cookies to improve the user experience. By using this website you consent to all cookies in accordance with our cookie policy.

The Nobel Prize in Physics 2011

Saul Perlmutter
Brian P. Schmidt
Adam G. Riess

Press release

English
English (pdf)
Swedish
Swedish (pdf)

4 October 2011

The Royal Swedish Academy of Sciences has decided to award the Nobel Prize in Physics for 2011

with one half to

Saul Perlmutter
The Supernova Cosmology Project
Lawrence Berkeley National Laboratory and University of California,
Berkeley, CA, USA

Figure 26: The press release material for 2011 Nobel Prize in Physics.