🔭 Assignment: Measuring Cosmological Parameters Using Type Ia Supernovae

In this assignment, you'll analyze observational data from the Pantheon+SH0ES dataset of Type Ia supernovae to measure the Hubble constant H_0 and estimate the age of the universe. You will:

- Plot the Hubble diagram (distance modulus vs. redshift)
- Fit a cosmological model to derive H_0 and Ω_m
- Estimate the age of the universe
- Analyze residuals to assess the model
- Explore the effect of fixing Ω_m
- Compare low-z and high-z results

Let's get started!

📦 Getting Started: Setup and Libraries

Before we dive into the analysis, we need to import the necessary Python libraries:

- `numpy` , `pandas` — for numerical operations and data handling
- `matplotlib` — for plotting graphs
- `scipy.optimize.curve_fit` and `scipy.integrate.quad` — for fitting cosmological models and integrating equations
- `astropy.constants` and `astropy.units` — for physical constants and unit conversions

Make sure these libraries are installed in your environment. If not, you can install them using:

```
pip install numpy pandas matplotlib scipy astropy
```

In [624…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.integrate import quad
from astropy.constants import c
from astropy import units as u
```

📂 Load the Pantheon+SH0ES Dataset

We now load the observational supernova data from the Pantheon+SH0ES sample. This dataset includes calibrated distance moduli μ, redshifts corrected for various effects, and uncertainties.

Instructions:

- Make sure the data file is downloaded from Pantheon dataset and available locally.
- We use `delim_whitespace=True` because the file is space-delimited rather than comma-separated.
- Commented rows (starting with `#`) are automatically skipped.

We will extract:

- `zHD` : Hubble diagram redshift
- `MU_SH0ES` : Distance modulus using SH0ES calibration
- `MU_SH0ES_ERR_DIAG` : Associated uncertainty

More detailed column names and the meanings can be referred here:

> Finally, we include a combined file of all the fitted parameters for each SN, before and after light-curve cuts are applied. This is in the format of a .FITRES file and has all the meta-information listed above along with the fitted SALT2 parameters. We show a screenshot of the release in Figure 7. Here, we give brief descriptions of each column. **CID** – name of SN. **CIDint** – counter of SNe in the sample. **IDSURVEY** – ID of the survey. **TYPE** – whether SN Ia or not – all SNe in this sample are SNe Ia. **FIELD** – if observed in a particular field. **CUTFLAG_SNANA** – any bits in light-curve fit flagged. **ERRFLAG_FIT** – flag in fit. **zHEL** – heliocentric redshift. **zHELERR** – heliocentric redshift error. **zCMB** – CMB redshift. **zCMBERR** – CMB redshift error. **zHD** – Hubble Diagram redshift. **zHDERR** – Hubble Diagram redshift error. **VPEC** – peculiar velocity. **VPECERR** – peculiar-velocity error. **MWEBV** – MW extinction. **HOST_LOGMASS** – mass of host. **HOST_LOGMASS_ERR** – error in mass of host. **HOST_sSFR** – sSFR of host. **HOST_sSFR_ERR** – error in sSFR of host. **PKMJDINI** – initial guess for PKMJD. **SNRMAX1** – First highest signal-to-noise ratio (SNR) of light curve. **SNRMAX2** – Second highest SNR of light curve. **SNRMAX3** – Third highest SNR of light curve. **PKMJD** – Fitted PKMJD. **PKMJDERR** –

In [625…

```
# Local file path
file_path = "Pantheon+SH0ES.dat"
```

```
# Load the file
df = pd.read_csv(file_path, delim_whitespace=True, comment='#',  header=None,)
# See structure
print(df.head())
print(len(df))
print(df.shape)
```

```
      0         1        2        3        4        5        6        7   \
0     CID  IDSURVEY       zHD   zHDERR     zCMB  zCMBERR     zHEL  zHELERR
1  2011fe        51  0.00122  0.00084  0.00122    2e-05  0.00082    2e-05
2  2011fe        56  0.00122  0.00084  0.00122    2e-05  0.00082    2e-05
3  2012cg        51  0.00256  0.00084  0.00256    2e-05  0.00144    2e-05
4  2012cg        56  0.00256  0.00084  0.00256    2e-05  0.00144    2e-05

         8                9         10               11         12  \
0  m_b_corr  m_b_corr_err_DIAG  MU_SH0ES  MU_SH0ES_ERR_DIAG  CEPH_DIST
1   9.74571            1.51621   28.9987            1.51645     29.177
2   9.80286            1.51723   29.0559            1.51747     29.177
3   11.4703           0.781906   30.7233           0.782372    30.8433
4   11.4919           0.798612   30.7449           0.799068    30.8433

             13               14         15         16         17        18  \
0  IS_CALIBRATOR  USED_IN_SH0ES_HF          c       cERR         x1     x1ERR
1              1                 0    -0.1076   0.04008  -0.548188   0.13373
2              1                 0  -0.032895  0.038463  -0.380481    0.0861
3              1                 0    0.10073  0.018231    0.49196  0.023545
4              1                 0    0.12247   0.03903    0.71261  0.083554

        19         20       21          22           23           24  \
0       mB      mBERR       x0       x0ERR      COV_x1_c     COV_x1_x0
1  9.58436  0.0327221  2.63181   0.0793177    0.00011378   -0.00052525
2  9.78448  0.0352442   2.1888   0.0710511  -0.000443845   -0.00150198
3  11.8161  0.0237119  0.33695  0.00735879  -6.06025e-05   9.35054e-06
4  11.8801  0.0359611  0.31765   0.0105211   0.000222656  -0.000136637

            25       26       27       28        29           30    31  \
0      COV_c_x0       RA      DEC  HOST_RA  HOST_DEC  HOST_ANGSEP  VPEC
1   -0.00272765  210.774  54.2737     -999      -999           -9     0
2   -0.00220084  210.774  54.2737     -999      -999           -9     0
3  -0.000110842  186.803   9.4203     -999      -999           -9     0
4  -0.000344022  186.803   9.4203     -999      -999           -9     0

        32          33            34               35       36        37  \
0  VPECERR       MWEBV  HOST_LOGMASS  HOST_LOGMASS_ERR    PKMJD  PKMJDERR
1      250  0.00758935        10.677               -9    55815    0.1071
2      250  0.00758935        10.677                0  55815.2    0.0579
3      250   0.0177724         9.633            0.002  56082.4    0.0278
4      250   0.0177724         9.633            0.004  56082.4    0.0667

     38       39          40              41               42  \
0  NDOF  FITCHI2      FITPROB  m_b_corr_err_RAW  m_b_corr_err_VPEC
1    36  26.8859      0.86447            0.0991             1.496
2   101  88.3064      0.81222            0.0971             1.496
3   165    233.5  0.000358347            0.0399            0.7134
4    55  100.122  0.000193186            0.0931            0.7134

           43             44                  45                 46
0  biasCor_m_b  biasCorErr_m_b  biasCor_m_b_COVSCALE  biasCor_m_b_COVADD
1       0.0381           0.005                     1               0.003
2      -0.0252           0.003                     1               0.004
3       0.0545           0.019                     1               0.036
4       0.0622           0.028                     1                0.04
1702
(1702, 47)
C:\Users\asus\AppData\Local\Temp\ipykernel_8580\3109395580.py:5: FutureWarning: The 'delim_whitespace' keyword in pd.read_cs
v is deprecated and will be removed in a future version. Use ``sep='\s+'`` instead
  df = pd.read_csv(file_path, delim_whitespace=True, comment='#',  header=None,)
```

📑 Preview Dataset Columns

Before diving into the analysis, let's take a quick look at the column names in the dataset. This helps us verify the data loaded correctly and identify the relevant columns we'll use for cosmological modeling.

In [626…

```
print(df[[2, 10, 11]].to_string())
```

```
          2         10                11
0        zHD  MU_SH0ES  MU_SH0ES_ERR_DIAG
1    0.00122   28.9987            1.51645
2    0.00122   29.0559            1.51747
3    0.00256   30.7233           0.782372
4    0.00256   30.7449           0.799068
5    0.00299   30.7757           0.881212
6    0.00317   30.7946           0.614535
7    0.00331   30.4604           0.594683
8    0.00331   30.5528           0.580251
9    0.00331   30.4013           0.578445
10   0.00331   30.5107           0.578546
11   0.00333   31.5011           0.591006
12   0.00349   31.7933            0.55272
13   0.00349   31.4967           0.545492
14   0.00359   31.6969           0.550993
15   0.00384   30.9999           0.566505
```

1645	0.73568	43.1807	0.25711
1646	0.73596	43.1675	0.425639
1647	0.73675	42.992	0.21489
1648	0.74196	43.4813	0.32514
1649	0.74276	43.0825	0.327179
1650	0.74573	43.1675	0.195179
1651	0.74607	42.9817	0.25474
1652	0.74911	42.9396	0.262488
1653	0.7493	43.1772	0.163225
1654	0.75074	42.9012	0.318716
1655	0.75899	43.2469	0.233153
1656	0.76076	43.0667	0.201161
1657	0.7611	43.1767	0.236628
1658	0.76165	43.131	0.239872
1659	0.76566	43.2369	0.225434
1660	0.76666	43.4224	0.250162
1661	0.76674	43.3623	0.27025
1662	0.76709	43.5587	0.25715
1663	0.76865	43.2929	0.311675
1664	0.76932	43.2168	0.162741
1665	0.77309	43.3568	0.295081
1666	0.77929	43.0395	0.311092
1667	0.78807	43.3042	0.232049
1668	0.78907	43.3837	0.271177
1669	0.78928	43.1995	0.165559
1670	0.79662	42.9026	0.372096
1671	0.79863	43.4338	0.288144
1672	0.83981	43.36	0.211845
1673	0.83981	43.2012	0.390423
1674	0.85482	43.6168	0.299913
1675	0.93585	43.5331	0.239939
1676	0.97423	44.1111	0.201519
1677	1.01242	44.1177	0.367085
1678	1.01988	44.2757	0.225903
1679	1.02088	43.9706	0.47547
1680	1.02789	44.5364	0.353385
1681	1.04817	44.4229	0.488758
1682	1.12092	44.4405	0.22363
1683	1.23225	44.3207	0.610496
1684	1.23597	44.905	0.273735
1685	1.29911	44.8641	0.275291
1686	1.3041	44.7206	0.316517
1687	1.30611	45.0197	0.445162
1688	1.31317	44.7867	0.316349
1689	1.3291	44.8745	0.238344
1690	1.34101	44.7541	0.306395
1691	1.35136	44.5492	0.31846
1692	1.35608	44.8094	0.238282
1693	1.39103	44.7944	0.374345
1694	1.41633	44.4579	0.735776
1695	1.5429	45.0902	0.371014
1696	1.54901	45.293	0.233771
1697	1.61505	45.1595	0.333024
1698	1.69706	45.2863	0.38048
1699	1.80119	45.4865	0.281981
1700	1.91165	45.4233	0.358642
1701	2.26137	46.1828	0.281309

🧹 Clean and Extract Relevant Data

To ensure reliable fitting, we remove any rows that have missing values in key columns:

- `zHD` : redshift for the Hubble diagram
- `MU_SH0ES` : distance modulus
- `MU_SH0ES_ERR_DIAG` : uncertainty in the distance modulus

We then extract these cleaned columns as NumPy arrays to prepare for analysis and modeling.

In [ ]:
```
# Filter for entries with usable data based on the required columns
z = df[2].values[1:].astype(float)   # skip header row and converted into float type
m = df[10].values[1:] .astype(float)
m_e = df[11].values[1:].astype(float)
```

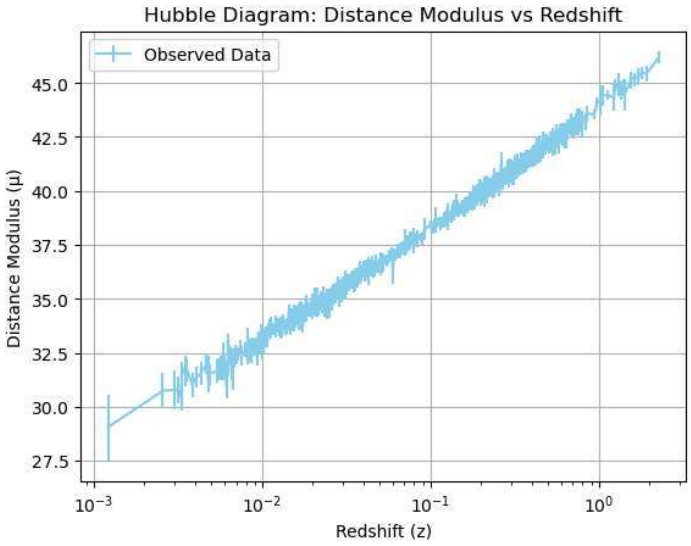
📈 Plot the Hubble Diagram

Let's visualize the relationship between redshift z and distance modulus μ, known as the Hubble diagram. This plot is a cornerstone of observational cosmology—it allows us to compare supernova observations with theoretical predictions based on different cosmological models.

We use a logarithmic scale on the redshift axis to clearly display both nearby and distant supernovae.

In [686…
```
# Write a code to plot the distance modulus and the redshift (x-axis), label them accordingly.
plt.figure()
plt.errorbar(z, m, m_e, label='Observed Data', color='skyblue')
plt.xlabel('Redshift (z)')
plt.ylabel('Distance Modulus (μ)')
plt.title('Hubble Diagram: Distance Modulus vs Redshift')
plt.grid(True)
```

```
#Try using log scale in x-axis
plt.xscale('log')
plt.legend()
plt.show()
```



Define the Cosmological Model

We now define the theoretical framework based on the flat $\Lambda\mathrm{CDM}$ model (read about the model in wikipedia if needed). This involves:

- The dimensionless Hubble parameter:

$$E(z) = \sqrt{\Omega_m (1+z)^3 + (1-\Omega_m)}$$

- The distance modulus is:

$$\mu(z) = 5 \log_{10}(d_L / \mathrm{Mpc}) + 25$$

- And the corresponding luminosity distance :

$$d_L(z) = (1+z) \cdot \frac{c}{H_0} \int_0^z \frac{dz'}{E(z')}$$

These equations allow us to compute the expected distance modulus from a given redshift z, Hubble constant H_0, and matter density parameter Ω_m.

In [ ]:
```
# Define the E(z) for flat LCDM
def E(z, Omega_m):
    return np.sqrt(Omega_m * (1+z)**3 + (1 - Omega_m))


# Luminosity distance in Mpc, try using scipy quad to integrate.
def luminosity_distance(z, H0, Omega_m):
    integral,_ = quad(lambda z_: 1 / E(z_, Omega_m), 0, z)
    d_L = (1 + z) * (c.to('km/s').value / H0) * integral
    return d_L  # in Mpc


# Theoretical distance modulus, use above function inside mu_theory to compute luminosity distance
def mu_theory(z, H0, Omega_m):
    return 5 * np.log10([luminosity_distance(z_, H0, Omega_m) for z_ in z]) + 25
```

Fit the Model to Supernova Data

We now perform a non-linear least squares fit to the supernova data using our theoretical model for $\mu(z)$. This fitting procedure will estimate the best-fit values for the Hubble constant H_0 and matter density parameter Ω_m, along with their associated uncertainties.

We'll use:

- `curve_fit` from `scipy.optimize` for the fitting.
- The observed distance modulus (\mu), redshift (z), and measurement errors.

The initial guess is:

- $H_0 = 70\,\mathrm{km/s/Mpc}$
- $\Omega_m = 0.3$

In [630…

```
# Initial guess: H0 = 70, Omega_m = 0.3
p0 = [70, 0.3]

# Write a code for fitting and taking error out of the parameters
p_optimal, covariance = curve_fit(mu_theory, z, m, p0=p0, sigma=m_e, absolute_sigma=True)
H0_fit, Omega_m_fit = p_optimal
H0_err, Omega_m_err = np.sqrt(np.diag(covariance))

print(f"Fitted H0 = {H0_fit:.2f} ± {H0_err:.2f} km/s/Mpc")
print(f"Fitted Omega_m = {Omega_m_fit:.3f} ± {Omega_m_err:.3f}")
```

```
Fitted H0 = 72.97 ± 0.26 km/s/Mpc
Fitted Omega_m = 0.351 ± 0.019
```

⏳ Estimate the Age of the Universe

Now that we have the best-fit values of H_0 and Ω_m we can estimate the age of the universe. This is done by integrating the inverse of the Hubble parameter over redshift:

$$t_0 = \int_0^\infty \frac{1}{(1+z)H(z)}\,dz$$

We convert H_0 to SI units and express the result in gigayears (Gyr). This provides an independent check on our cosmological model by comparing the estimated age to values from other probes like Planck CMB measurements.

In [687…

```
# Write the function for age of the universe as above

def age_of_universe(H0, Omega_m):
    integral = lambda z_: 1 / ((1 + z_) * E(z_, Omega_m))
    integral, _ = quad(integral, 0, np.inf)  # Using quad function to integrate the equation
    H0_si = H0 * u.km / (u.s * u.Mpc)  # Convert H0 to SI units
    t0 = integral / H0_si.to('1/s').value / (3600 * 24 * 365.25 * 1e9)  # Convert to Gyr
    return t0 # in Gyr

t0 = age_of_universe(H0_fit, Omega_m_fit)
print(f"Estimated age of Universe: {t0:.2f} Gyr")
```

```
Estimated age of Universe: 12.36 Gyr
```

📊 Analyze Residuals

To evaluate how well our cosmological model fits the data, we compute the residuals:

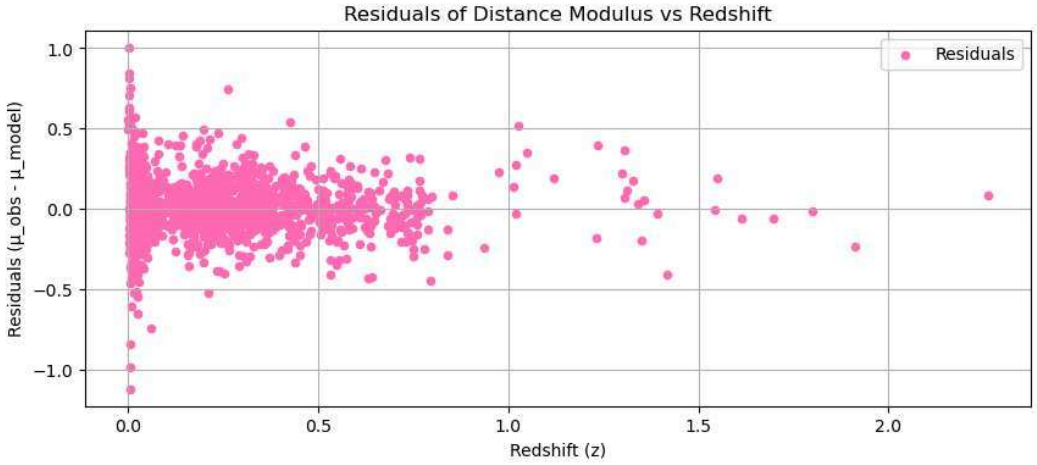
$$\text{Residual} = \mu_{\text{obs}} - \mu_{\text{model}}$$

Plotting these residuals against redshift helps identify any systematic trends, biases, or outliers. A good model fit should show residuals scattered randomly around zero without any significant structure.

In [632…

```
# Write the code to find residual by computing mu_theory and then plot
mu_model = mu_theory(z, H0_fit, Omega_m_fit)  # Compute the theoretical distance modulus

# Calculate residuals
residuals = m - mu_model
# Plot residuals
plt.figure(figsize=(10, 4))
plt.grid(True)
plt.scatter(z, residuals, label='Residuals', color='hotpink', s=20)
plt.legend()
plt.xlabel('Redshift (z)')
plt.ylabel('Residuals (μ_obs - μ_model)')
plt.title('Residuals of Distance Modulus vs Redshift')
plt.show()
```



🔧 Fit with Fixed Matter Density

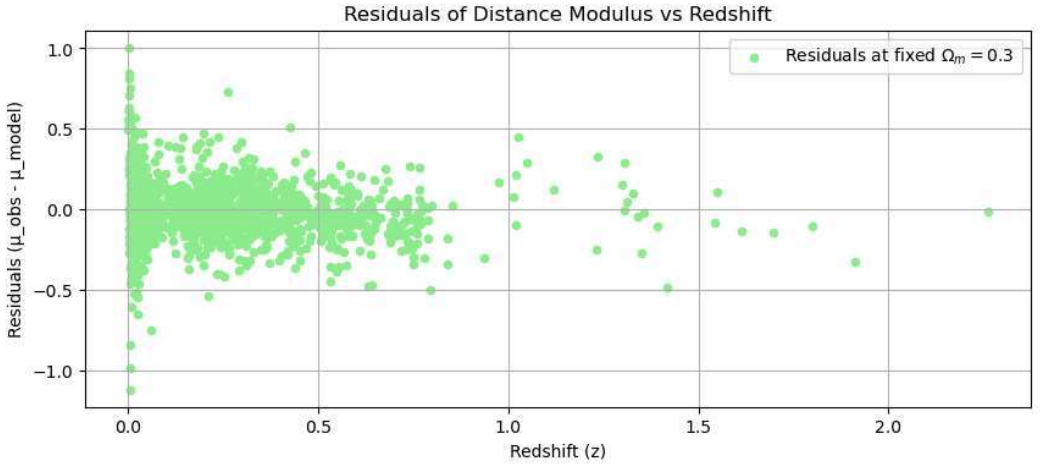
To reduce parameter degeneracy, let's fix $\Omega_m = 0.3$ and fit only for the Hubble constant H_0.

In [688…

```
def mu_fixed_Om(z, H0):
    return mu_theory(z, H0, Omega_m=0.3)
residuals = m - mu_fixed_Om(z, H0_fit)

# Try fitting with this fixed value 'Omega_m=0.3'
plt.figure(figsize=(10, 4))
plt.grid(True)
plt.scatter(z, residuals, label='Residuals at fixed $ \Omega_m = 0.3 $ ', color='lightgreen', s=20)
plt.legend()
plt.xlabel('Redshift (z)')
plt.ylabel('Residuals (μ_obs - μ_model)')
plt.title('Residuals of Distance Modulus vs Redshift')
plt.show()
```

```
<>:8: SyntaxWarning: invalid escape sequence '\O'
<>:8: SyntaxWarning: invalid escape sequence '\O'
C:\Users\asus\AppData\Local\Temp\ipykernel_8580\263392893.py:8: SyntaxWarning: invalid escape sequence '\O'
  plt.scatter(z, residuals, label='Residuals at fixed $ \Omega_m = 0.3 $ ', color='lightgreen', s=20)
```



🔍 Compare Low-z and High-z Subsamples

Finally, we examine whether the inferred value of H_0 changes with redshift by splitting the dataset into:

- **Low-z** supernovae ($z < 0.1$)
- **High-z** supernovae ($z \geq 0.1$)

We then fit each subset separately (keeping $\Omega_m = 0.3$) to explore any potential tension or trend with redshift.

In [ ]:

```
# Split the data for the three columns and do the fitting again and see
z_split = 0.1

# Calculate H0 for low and high redshift by using curve_fit function
H0_low, cov_low = curve_fit(mu_theory, z[z < z_split], m[z < z_split], p0=p0, sigma=m_e[z < z_split], absolute_sigma=True)
H0_high, cov_high = curve_fit(mu_theory, z[z >= z_split], m[z >= z_split], p0=p0, sigma=m_e[z >= z_split], absolute_sigma=True)

# Calculate the errors(standard deviation) for the low and high redshift
H0_low_err = np.sqrt(cov_low[0][0])
H0_high_err = np.sqrt(cov_high[0][0])

# print the results
print(f"Low-z (z < {z_split}): H₀ = {H0_low[0]:.2f} ± {H0_low_err:.2f} km/s/Mpc")
print(f"High-z (z ≥ {z_split}): H₀ = {H0_high[0]:.2f} ± {H0_high_err:.2f} km/s/Mpc")
```

```
Low-z (z < 0.1): H₀ = 72.74 ± 0.59 km/s/Mpc
High-z (z ≥ 0.1): H₀ = 73.18 ± 0.50 km/s/Mpc
```

You can check your results and potential reasons for different values from accepted constant using this paper by authors of the Pantheon+ dataset

You can find more about the dataset in the paper too

In [677…

```
# Hubble diagram with model fit
plt.figure()
plt.errorbar(z, m, m_e ,color='orange', label='Observed Data')
mu_model = mu_theory(z, H0_fit, Omega_m_fit)
plt.plot(z, mu_model, label='Model Fit', color='black')
plt.grid(True)
```

```
plt.xlabel('Redshift (z)')
plt.ylabel('Distance Modulus')
plt.title('Hubble diagram with Model fit')
plt.xscale('log')
plt.legend()
plt.show()
```