2_hubble_parameter

July 1, 2025

1 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!

1.1 DONE BY GAUTHAM R

1.2 Application number 561440

1.3 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:

"'bash pip install numpy pandas matplotlib scipy astropy

[2]:
```
!pip install numpy pandas matplotlib scipy astropy
```

```
Requirement already satisfied: numpy in c:\python313\lib\site-packages (2.1.3)
Requirement already satisfied: pandas in c:\python313\lib\site-packages (2.2.3)
Requirement already satisfied: matplotlib in c:\python313\lib\site-packages
(3.9.2)
Requirement already satisfied: scipy in c:\python313\lib\site-packages (1.15.3)
Collecting astropy
  Downloading astropy-7.1.0-cp313-cp313-win_amd64.whl.metadata (10 kB)
```

```
Requirement already satisfied: python-dateutil>=2.8.2 in c:\python313\lib\site-
packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\python313\lib\site-packages
(from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\python313\lib\site-packages
(from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\python313\lib\site-
packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\python313\lib\site-packages
(from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\python313\lib\site-
packages (from matplotlib) (4.55.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\python313\lib\site-
packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in c:\python313\lib\site-packages
(from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in c:\python313\lib\site-packages (from
matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\python313\lib\site-
packages (from matplotlib) (3.2.0)
Collecting pyerfa>=2.0.1.1 (from astropy)
  Downloading pyerfa-2.0.1.5-cp39-abi3-win_amd64.whl.metadata (5.9 kB)
Collecting astropy-iers-data>=0.2025.4.28.0.37.27 (from astropy)
  Downloading astropy_iers_data-0.2025.6.30.0.39.40-py3-none-any.whl.metadata
(3.4 kB)
Requirement already satisfied: PyYAML>=6.0.0 in c:\python313\lib\site-packages
(from astropy) (6.0.2)
Requirement already satisfied: six>=1.5 in c:\python313\lib\site-packages (from
python-dateutil>=2.8.2->pandas) (1.16.0)
Downloading astropy-7.1.0-cp313-cp313-win_amd64.whl (6.3 MB)
   ---------------------------------------- 0.0/6.3 MB ? eta -:--:--
   -------------- ------------------------ 2.4/6.3 MB 12.8 MB/s eta 0:00:01
   -------------------------------- ------ 5.2/6.3 MB 12.8 MB/s eta 0:00:01
   ---------------------------------------- 6.3/6.3 MB 11.4 MB/s eta 0:00:00
Downloading astropy_iers_data-0.2025.6.30.0.39.40-py3-none-any.whl (2.0 MB)
   ---------------------------------------- 0.0/2.0 MB ? eta -:--:--
   ---------------------------------------- 2.0/2.0 MB 11.3 MB/s eta 0:00:00
Downloading pyerfa-2.0.1.5-cp39-abi3-win_amd64.whl (349 kB)
Installing collected packages: pyerfa, astropy-iers-data, astropy
Successfully installed astropy-7.1.0 astropy-iers-data-0.2025.6.30.0.39.40
pyerfa-2.0.1.5


[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

[3]:
```
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
```

1.4 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.

1.4.1 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** –

[4]:
```
# Local file path
file_path = "Pantheon+SH0ES.dat"

# Load the file
df = pd.read_csv(file_path, delim_whitespace=True, comment="#")

# See structure
df.head()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_9736\1812532777.py:5: FutureWarning:
The 'delim_whitespace' keyword in pd.read_csv is deprecated and will be removed
in a future version. Use ``sep='\s+'`` instead
  df = pd.read_csv(file_path, delim_whitespace=True, comment="#")
```

[4]:
```
              CID  IDSURVEY      zHD   zHDERR     zCMB  zCMBERR     zHEL  \
0         2011fe        51  0.00122  0.00084  0.00122  0.00002  0.00082
1         2011fe        56  0.00122  0.00084  0.00122  0.00002  0.00082
2         2012cg        51  0.00256  0.00084  0.00256  0.00002  0.00144
3         2012cg        56  0.00256  0.00084  0.00256  0.00002  0.00144
4  1994DRichmond        50  0.00299  0.00084  0.00299  0.00004  0.00187

   zHELERR  m_b_corr  m_b_corr_err_DIAG  …  PKMJDERR  NDOF   FITCHI2  \
0  0.00002   9.74571           1.516210  …    0.1071    36   26.8859
1  0.00002   9.80286           1.517230  …    0.0579   101   88.3064
2  0.00002  11.47030           0.781906  …    0.0278   165  233.5000
3  0.00002  11.49190           0.798612  …    0.0667    55  100.1220
4  0.00004  11.52270           0.880798  …    0.0522   146  109.8390

    FITPROB  m_b_corr_err_RAW  m_b_corr_err_VPEC  biasCor_m_b  biasCorErr_m_b  \
0  0.864470            0.0991             1.4960       0.0381           0.005
1  0.812220            0.0971             1.4960      -0.0252           0.003
2  0.000358            0.0399             0.7134       0.0545           0.019
3  0.000193            0.0931             0.7134       0.0622           0.028
4  0.988740            0.0567             0.6110       0.0650           0.009

   biasCor_m_b_COVSCALE  biasCor_m_b_COVADD
0                   1.0               0.003
1                   1.0               0.004
2                   1.0               0.036
3                   1.0               0.040
4                   1.0               0.006

[5 rows x 47 columns]
```

1.5 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.

[5]:
```
print("Available columns:\n")
print(df.columns.tolist())
```

```
Available columns:

['CID', 'IDSURVEY', 'zHD', 'zHDERR', 'zCMB', 'zCMBERR', 'zHEL', 'zHELERR',
'm_b_corr', 'm_b_corr_err_DIAG', 'MU_SHOES', 'MU_SHOES_ERR_DIAG', 'CEPH_DIST',
'IS_CALIBRATOR', 'USED_IN_SHOES_HF', 'c', 'cERR', 'x1', 'x1ERR', 'mB', 'mBERR',
'x0', 'x0ERR', 'COV_x1_c', 'COV_x1_x0', 'COV_c_x0', 'RA', 'DEC', 'HOST_RA',
'HOST_DEC', 'HOST_ANGSEP', 'VPEC', 'VPECERR', 'MWEBV', 'HOST_LOGMASS',
'HOST_LOGMASS_ERR', 'PKMJD', 'PKMJDERR', 'NDOF', 'FITCHI2', 'FITPROB',
'm_b_corr_err_RAW', 'm_b_corr_err_VPEC', 'biasCor_m_b', 'biasCorErr_m_b',
```

```
'biasCor_m_b_COVSCALE', 'biasCor_m_b_COVADD']
```

1.6 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_SHOES`: distance modulus
- `MU_SHOES_ERR_DIAG`: uncertainty in the distance modulus

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

[6]:
```
# Filter for entries with usable data based on the required columns
missing_counts = df.isnull().sum()
print(missing_counts[missing_counts > 0])
```

```
Series([], dtype: int64)
```

[7]:
```
df_clean = df.dropna(subset=["zHD", "MU_SHOES", "MU_SHOES_ERR_DIAG"])
```

[8]:
```
z = df_clean["zHD"].values
mu = df_clean["MU_SHOES"].values
mu_err = df_clean["MU_SHOES_ERR_DIAG"].values

# Preview the number of usable entries
print(f"Number of usable supernovae: {len(z)}")
```

```
Number of usable supernovae: 1701
```

[ ]:
```

```

1.7 Plot the Hubble Diagram

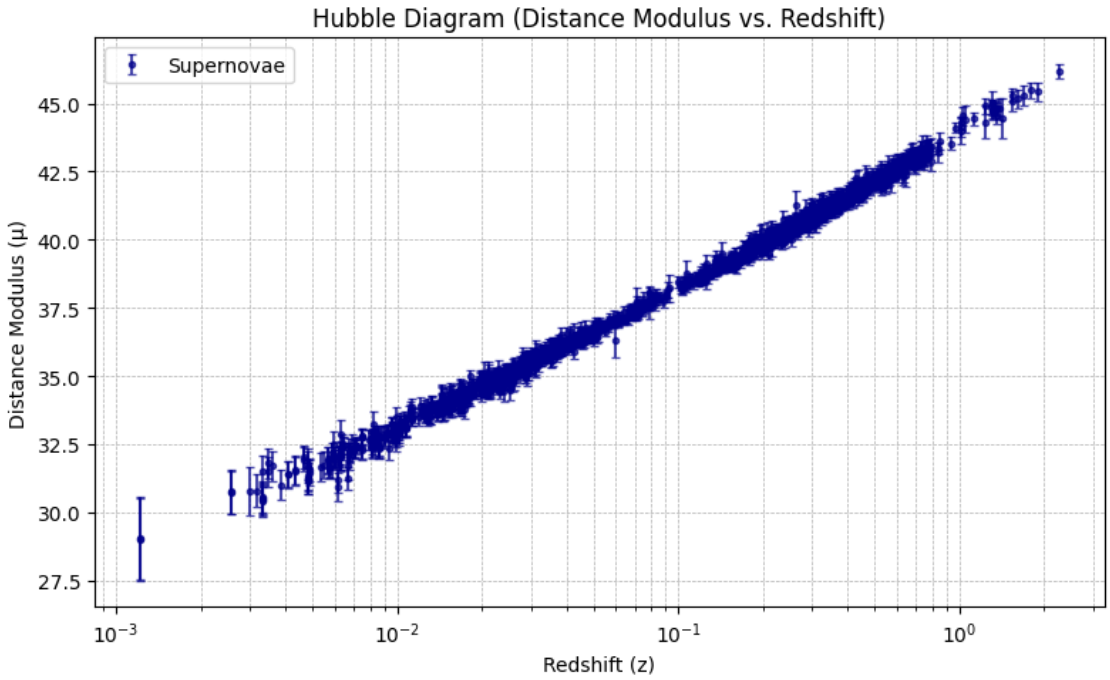
Let's visualize the relationship between redshift z and distance modulus μ, known as the Hubble diagram. This plot is a cornerstone of observational cosmology—it allows us to compare supernova observations with theoretical predictions based on different cosmological models.

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

[9]:
```
# Write a code to plot the distance modulus and the redshift (x-axis), label␣
 ↪them accordingly.
plt.figure(figsize=(8, 5))
plt.errorbar(z, mu, yerr=mu_err, fmt='o', markersize=3, capsize=2,␣
 ↪color='darkblue', alpha=0.7, label='Supernovae')

#Try using log scale in x-axis
plt.xscale('log')
plt.xlabel("Redshift (z)")
plt.ylabel("Distance Modulus ( )")
```

```
plt.title("Hubble Diagram (Distance Modulus vs. Redshift)")
plt.grid(True, which="both", ls="--", lw=0.5)
plt.legend()
plt.tight_layout()
plt.show()
```



1.8 Define the Cosmological Model

We now define the theoretical framework based on the flat ΛCDM 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:

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

- 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.

[10]:
```
# 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 = np.array([quad(lambda z_: 1/E(z_, Omega_m), 0, zi)[0] for zi in␣
 ↪z])
    dL = (1 + z) * (c.to('km/s').value / H0) * integral
    return dL

# Theoretical distance modulus, use above function inside mu_theory to compute␣
 ↪luminosity distance
def mu_theory(z, H0, Omega_m):
    dL = luminosity_distance(z, H0, Omega_m)  # in Mpc
    return 5 * np.log10(dL) + 25
```

1.9 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 ( ), redshift (z), and measurement errors.

The initial guess is: - $ H_0 = 70 $, km/s/Mpc $ - $\Omega_m = 0.3$

[11]:
```
# Initial guess: H0 = 70, Omega_m = 0.3
p0 = [70, 0.3]

# Write a code for fitting and taking error out of the parameters
popt, pcov = curve_fit(mu_theory, z, mu, sigma=mu_err, p0=p0,␣
 ↪absolute_sigma=True)
H0_fit, Omega_m_fit = popt
H0_err, Omega_m_err = np.sqrt(np.diag(pcov))

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
```

1.10 Estimate the Age of the Universe

Now that we have the best-fit values of $ H_0 $ and $ _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.

[13]:
```
# Write the function for age of the universe as above

def age_of_universe(H0, Omega_m):
    H0_SI = H0 * 1e3 / (3.086e22)  # Convert km/s/Mpc to 1/s
    # Integrand for age equation
    def integrand(z):
        return 1 / ((1 + z) * np.sqrt(Omega_m * (1 + z)**3 + (1 - Omega_m)))

    # Integrate from z = 0 to ∞
    integral, _ = quad(integrand, 0, np.inf)

    age_sec = integral / H0_SI
    age_gyr = age_sec / (3600 * 24 * 365.25 * 1e9)  # Convert seconds to Gyr
    return age_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
```

1.11 Analyze Residuals

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

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

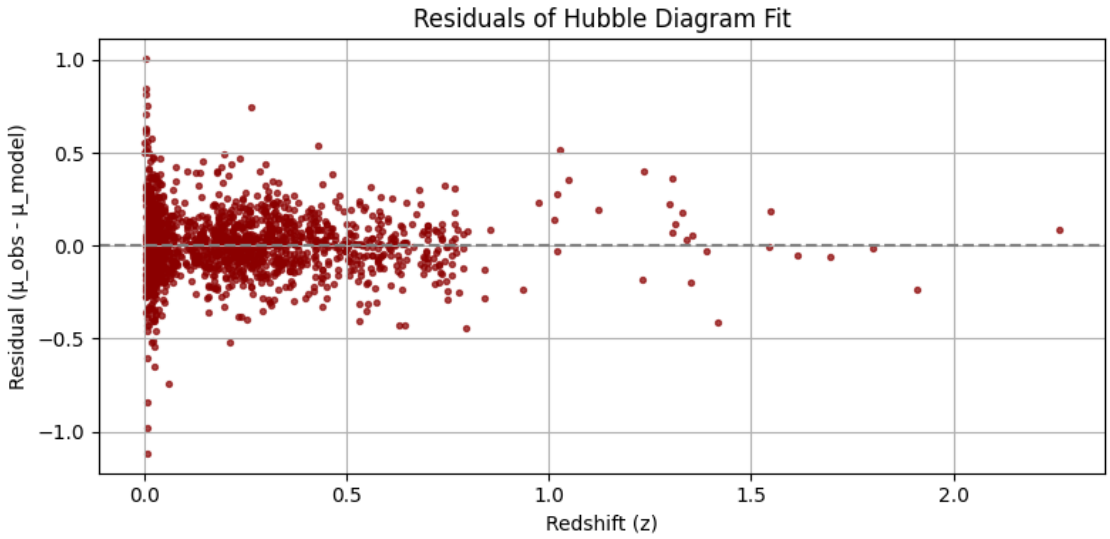
Plotting these residuals against redshift helps identify any systematic trends, biases, or outliers. A good model fit should show residuals scattered randomly around zero without any significant structure.

[14]:
```
# Write the code to find residual by computing mu_theory and then plot
mu_model = mu_theory(z, H0_fit, Omega_m_fit)

# Compute residuals
residuals = mu - mu_model

# Plot residuals
plt.figure(figsize=(8, 4))
plt.scatter(z, residuals, s=8, color="darkred", alpha=0.7)
plt.axhline(0, color="gray", linestyle="--")
plt.xlabel("Redshift (z)")
```

```
plt.ylabel("Residual ( _obs -  _model)")
plt.title("Residuals of Hubble Diagram Fit")
plt.grid(True)
plt.tight_layout()
plt.show()
```



1.12 Fit with Fixed Matter Density

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

[15]:
```
def mu_fixed_Om(z, H0):
    return mu_theory(z, H0, Omega_m=0.3)

# Try fitting with this fixed value
popt_fixed, pcov_fixed = curve_fit(mu_fixed_Om, z, mu, sigma=mu_err, p0=[70],␣
 ↪absolute_sigma=True)
H0_fixed = popt_fixed[0]
H0_fixed_err = np.sqrt(np.diag(pcov_fixed))[0]

print(f"Fit with fixed Omega_m = 0.3: H0 = {H0_fixed:.2f} ± {H0_fixed_err:.2f}␣
 ↪km/s/Mpc")
```

```
Fit with fixed Omega_m = 0.3: H0 = 73.53 ± 0.17 km/s/Mpc
```

1.13 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 0.1 $)

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

[16]:
```
# Split the data for the three columns and do the fitting again and see
z_split = 0.1

# Low-z sample
z_low = z[z < z_split]
mu_low = mu[z < z_split]
mu_err_low = mu_err[z < z_split]

# High-z sample
z_high = z[z >= z_split]
mu_high = mu[z >= z_split]
mu_err_high = mu_err[z >= z_split]

# Fit H0 for both (with Omega_m fixed = 0.3)
H0_low, _ = curve_fit(mu_fixed_Om, z_low, mu_low, sigma=mu_err_low, p0=[70],␣
↪absolute_sigma=True)
H0_high, _ = curve_fit(mu_fixed_Om, z_high, mu_high, sigma=mu_err_high,␣
↪p0=[70], absolute_sigma=True)

print(f"Low-z (z < {z_split}): H  = {H0_low[0]:.2f} km/s/Mpc")
print(f"High-z (z   {z_split}): H  = {H0_high[0]:.2f} km/s/Mpc")
```

```
Low-z (z < 0.1): H  = 73.01 km/s/Mpc
High-z (z   0.1): H  = 73.85 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