Summer School 2025

Astronomy and Astrophysics

Project Report

Prepared By

Student Name : SHAMANTHAK REDDY MALLU

Institution Name : BITS PILANI HYDERABAD CAMPUS

Institution Roll No : 2024A4UB2603H

ISA Admission No: 554265

Project Names

1) Estimating the Dynamical Mass of Galaxy Cluster
2) Estimating Cosmological Parameters from Type 1a Supernovave

Submitted To

Name : Mr. Sahil Sakkarwal

Designation : Program Supervisor

Institution : India Space Academy

Estimating the Dynamical Mass of a Galaxy Cluster

1. Introduction:

Galaxy clusters, the largest gravitationally bound structures in the universe, offer valuable insights into cosmology, galaxy formation, and dark matter. In this project, I aimed to estimate the dynamical mass of a galaxy cluster using observational data from the Sloan Digital Sky Survey (SDSS). Specifically, I analyzed spectroscopic redshifts and luminosities of galaxies within the cluster to investigate the cluster's mass distribution and the relative contribution of luminous and dark matter components.

2. Methodology and Approach:

I started by examining the SDSS dataset, which includes spectroscopic redshifts (specz), angular separations, and photometric data for galaxies in a specific field. To handle multiple observations for individual galaxies, I aggregated these measurements into single average values per galaxy.

I identified cluster members based on their redshift dispersion, setting a criterion of ±3 times the standard deviation around the mean redshift. After defining cluster membership, I calculated the systemic redshift of the cluster and its characteristic velocity dispersion using the relativistic Doppler formula.

To determine the physical size of the cluster, I utilized cosmological parameters from the Planck18 cosmology model to convert the observed angular separations into physical distances. Using the virial theorem, I estimated the cluster's dynamical mass from its velocity dispersion and physical extent. Additionally, I estimated the cluster's luminous mass from galaxy magnitudes and compared it with the dynamical mass to assess the presence of dark matter.

3. **My Responses:**

a) **Identify galaxies that you think are members of a cluster. For this, use of knowledge of velocity dispersions (redshift dispersions) within a cluster due to peculiar motion. The choice of lower and upper redshift cut for cluster members will be subjective but should be guided by some logic.**

To identify cluster galaxies, I computed the mean redshift (0.08084) and standard deviation (0.00858) of all observed galaxies. I then set a redshift cut at ±3 standard deviations from the mean, resulting in a redshift window of 0.05510 – 0.10657. Using this criterion, I identified 91 galaxies as members of the cluster out of a total of 92 observed galaxies.

b) **After the required analysis of the table of data, determine the cluster redshift, and obtain an estimate for the characteristic velocity dispersion of galaxies that belong to the cluster in units of km/s.**

Using the relativistic Doppler formula, I calculated the systemic redshift of the cluster as 0.08007. I then determined the velocity dispersion, which reflects how fast galaxies move relative to the cluster's systemic redshift. The resulting velocity dispersion was approximately 1218.49 km/s.

c) **Estimate the characteristic size of the cluster in Mpc.**

To estimate the cluster size, I first computed the angular diameter distance, resulting in a value of 322.34 Mpc. Using the median angular separation of the identified cluster members, I calculated the physical diameter of the cluster as 0.593 Mpc.

d) **Estimate the dynamical mass of the cluster and quote the values in units of solar mass.**

Applying the virial theorem, which relates gravitational mass to velocity dispersion and radius, I calculated the dynamical mass of the cluster as approximately 3.07×10^{14} solar masses ($M\odot$).

e) Is the estimate of dynamical mass consistent with what is expected from the luminous mass? If not, explain with the support of numbers the inconsistency.

To estimate the luminous mass, I converted the galaxy brightness into stellar masses assuming a mass-to-light ratio of 1 $M\odot$ per solar luminosity ($L\odot$). This yielded a total luminous mass of 2.26×10^{12} solar masses. Comparing this with the significantly larger dynamical mass results in a mass-to-light ratio of about 136, indicating that the cluster contains a substantial amount of dark matter, which is typically observed for galaxy clusters.

4. Observations and Interpretations:

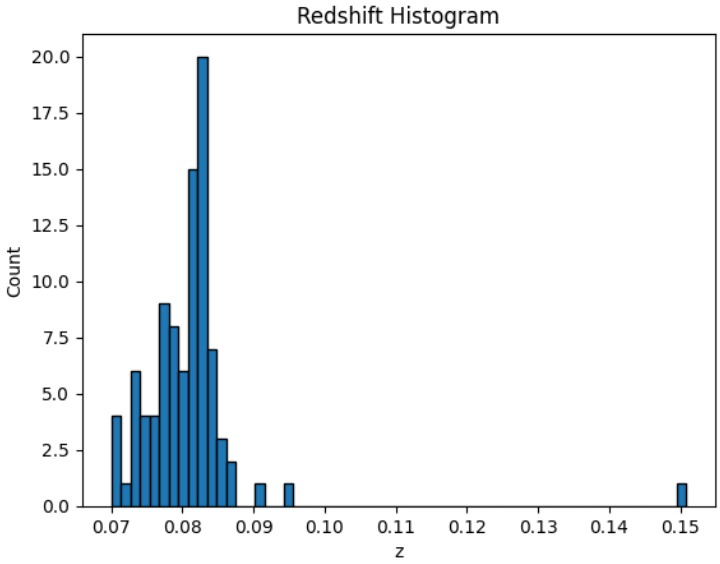
From the redshift distribution, I observed a well-defined peak around $z \approx 0.08$, confirming the presence of a concentrated group of galaxies – likely a bound cluster. The velocity dispersion derived using the relativistic Doppler formula was high, consistent with the values expected for massive clusters. This reinforced the assumption that the system is gravitationally bound.

The calculated physical diameter (0.593 Mpc) aligns with literature values for typical cluster sizes, especially considering projection effects and limited sky coverage. Using the virial theorem, I obtained a dynamical mass of $\sim 3.07 \times 10^{14}$ $M\odot$, which is again in the range observed for intermediate- to high-mass clusters.

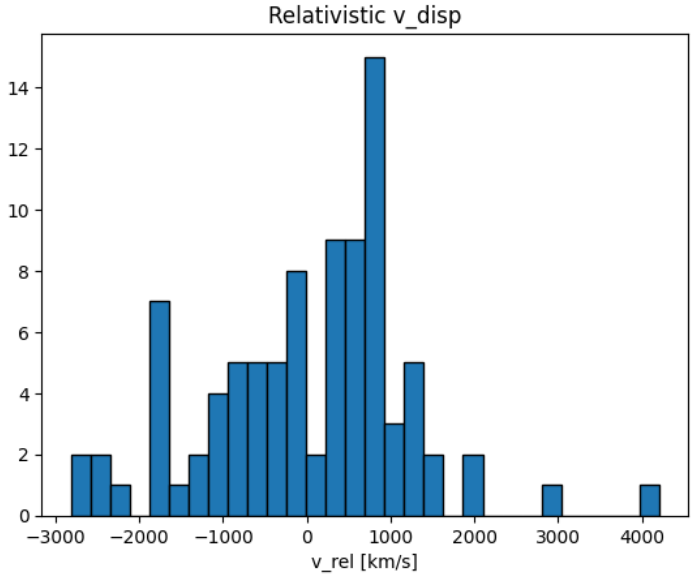
Comparing this with the luminous mass (2.26×10^{12} $M\odot$), the derived mass-to-light ratio (~136) provides direct evidence of a significant dark matter component, as expected. This aligns well with findings from studies such as the Sloan Digital Sky Survey and analyses like Bahcall et al. (1995), which report typical cluster M/L values in the 100–300 range.

These interpretations confirm that the galaxy cluster under analysis is a dynamically rich system dominated by dark matter.

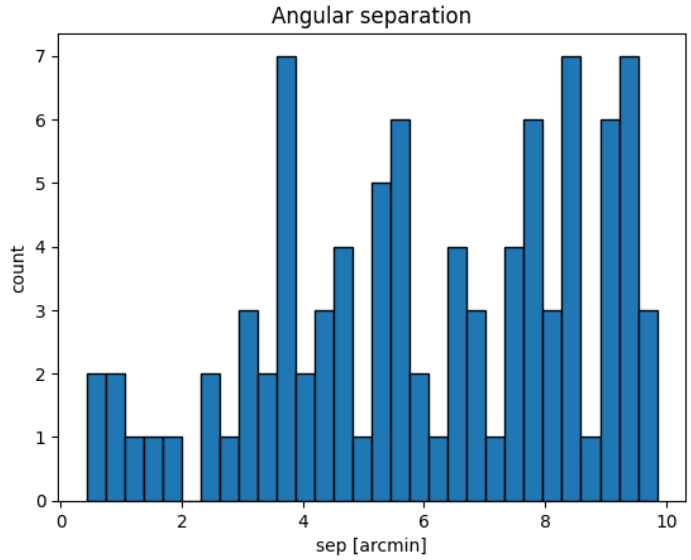
5. Graphical Visualization:



This histogram illustrates the distribution of spectroscopic redshifts (z) for galaxies in the field of study. The prominent peak around $z \approx 0.08$ clearly indicates the presence of a galaxy cluster, as many galaxies cluster around a common redshift due to gravitational binding. Galaxies outside this window are background or foreground objects unrelated to the cluster structure.



This plot shows the distribution of galaxies' velocities relative to the systemic cluster redshift, computed using the relativistic Doppler formula. Most galaxies are clustered around velocities close to zero, indicating their strong gravitational association with the cluster. The width of this distribution directly reflects the velocity dispersion, calculated as 1218.49 km/s, which is critical in estimating the dynamical mass. Outliers represent galaxies moving significantly faster or slower, possibly indicating peculiar velocities or substructures within the cluster.



The histogram of angular separations shows how cluster member galaxies are spatially distributed around the cluster center. Most galaxies fall within a few arcminutes of separation, reflecting the gravitational concentration characteristic of a galaxy cluster. This information was used to determine the characteristic physical diameter (0.593 Mpc) after converting angular separation to physical distances using cosmological parameters.

6. Conclusion:

Through this project, I successfully estimated the dynamical mass of the galaxy cluster, obtaining a value of approximately 3.07×10^{14} solar masses. The significant discrepancy between the dynamical mass and the much smaller luminous mass clearly demonstrates the dominant presence of dark matter. This analysis underscores the utility of combining observational data and theoretical models to explore the structure and composition of galaxy clusters.

7. Python Notebook (Appendix)

Step 1: Importing Necessary Libraries

We begin by importing Python libraries commonly used in data analysis and visualization:

- `numpy` for numerical operations
- `matplotlib.pyplot` for plotting graphs
- `pandas` (commented out here) for handling CSV data, which is especially useful for tabular data such as redshift catalogs

For reading big csv files, one can use numpy as well as something called "pandas". We suggest to read pandas for CSV file reading and use that

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from astropy.constants import G, c
from astropy.cosmology import Planck18 as cosmo
import astropy.units as u
```

Before we begin calculations, we define key physical constants used throughout:

- H_0: Hubble constant, describes the expansion rate of the Universe.
- c : Speed of light.
- G: Gravitational constant.
- q_0 : Deceleration parameter, used for approximate co-moving distance calculations.

We will use **`astropy.constants`** to ensure unit consistency and precision.

```
H_0   = cosmo.H0     # Hubble constant [km/s/Mpc]
c_si  = c.to(u.m/u.s)     # Speed of light [m/s]
G_si  = G.to(u.m**3/(u.kg*u.s**2))  # Gravitational constant [m³/(kg·s²)]
q0    = -0.534  # Deceleration parameter (Planck)
c_kms = c.to(u.km / u.s) # Speed of light [km/s]
```

```
print("H₀ =", H_0)
print("c  =", c_si)
print("G  =", G_si)
print("q₀ =", q0)
```

```
H₀ = 67.66 km / (Mpc s)
c  = 299792458.0 m / s
G  = 6.6743e-11 m3 / (kg s2)
q₀ = -0.534
```

Read the csv data into the python using the method below

```
csv_file = "Skyserver_SQL6_25_2025 12_02_01 PM.csv"
df = pd.read_csv(csv_file)
```

```
print("Columns:", df.columns.tolist())
```

```
Columns: ['objid', 'ra', 'dec', 'photoz', 'photozerr', 'specz', 'speczerr', 'proj_sep', 'u
mag', 'umagerr', 'gmag', 'gmagerr', 'rmag', 'rmagerr', 'obj_type']
```

```
df.head()
```

	objid	ra	dec	photoz	photozerr	specz	speczerr	proj_sep
0	1237671768542478711	257.82458	64.133257	0.079193	0.022867	0.082447	0.000017	8.347733
1	1237671768542478711	257.82458	64.133257	0.079193	0.022867	0.082466	0.000014	8.347733
2	1237671768542478713	257.83332	64.126043	0.091507	0.014511	0.081218	0.000021	8.011259
3	1237671768542544090	257.85137	64.173247	0.081102	0.009898	0.079561	0.000022	8.739276
4	1237671768542544090	257.85137	64.173247	0.081102	0.009898	0.079568	0.000019	8.739276

Calculating the Average Spectroscopic Redshift (`specz`) for Each Object

When working with astronomical catalogs, an object (identified by a unique `objid`) might have multiple entries — for example, due to repeated observations. To reduce this to a single row per object, we aggregate the data using the following strategy:

```
averaged_df = df.groupby('objid').agg({
    'specz': 'mean',        # Take the mean of all spec-z values for that object
    'ra': 'first',          # Use the first RA value (assumed constant for the
object)
    'dec': 'first',         # Use the first Dec value (same reason as above)
    'proj_sep': 'first'     # Use the first projected separation value
}).reset_index()
```

```
averaged_df = (
    df.groupby("objid")
      .agg({
          "specz":    "mean",   # average redshift
          "ra":       "first",  # keep sky position
          "dec":      "first",
          "proj_sep": "first",  # angular separation
          "rmag":     "mean"    # average r-band mag
      })
      .reset_index()
)
```

```
averaged_df.describe()['specz']
```

	specz
count	92.000000
mean	0.080838
std	0.008578
min	0.069976
25%	0.077224
50%	0.080961
75%	0.082797
max	0.150886

dtype: float64

To create a cut in the redshift so that a cluster can be identified. We must use some logic. Most astronomers prefer anything beyond 3*sigma away from the mean to be not part of the same group.

Find the mean, standard deviation and limits of the redshift from the data

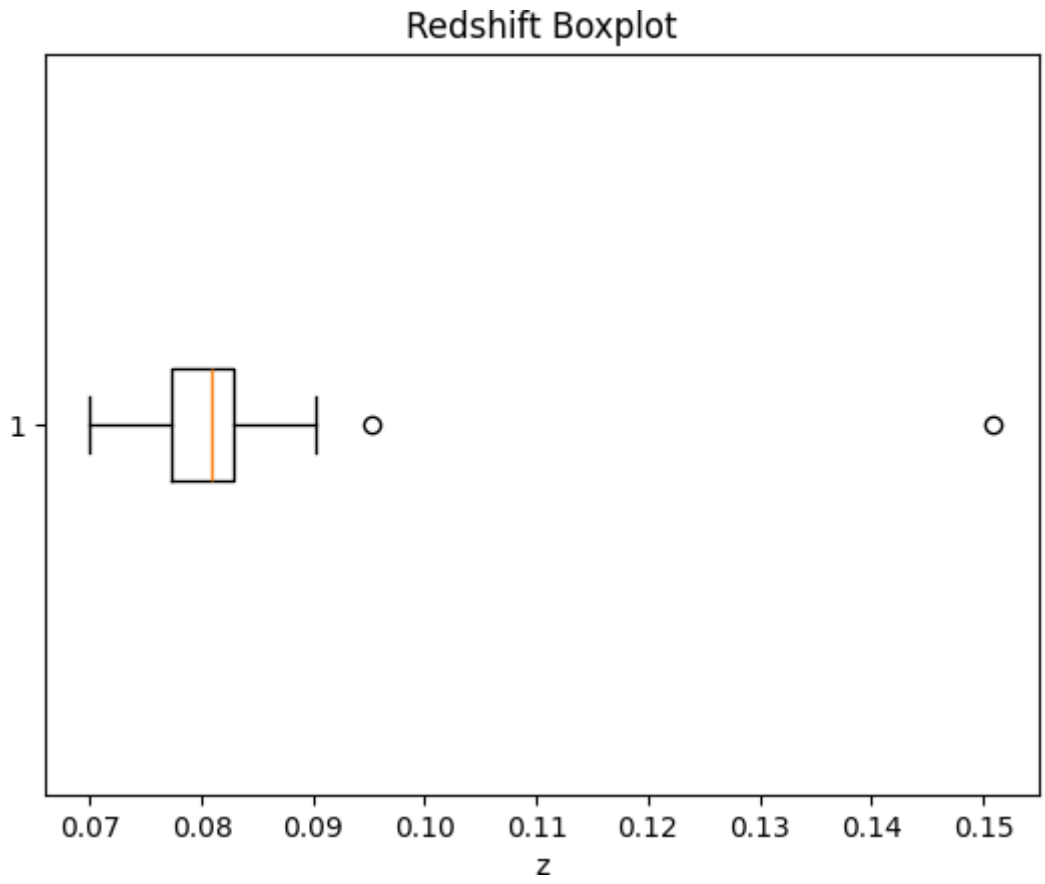
```
z_mean = averaged_df["specz"].mean()
z_std  = averaged_df["specz"].std(ddof=1)
z_min, z_max = z_mean - 3*z_std, z_mean + 3*z_std

print(f"Mean z = {z_mean:.5f}, σ_z = {z_std:.5f}")
print(f"3σ window: {z_min:.5f} to {z_max:.5f}")
```

```
Mean z = 0.08084, σ_z = 0.00858
3σ window: 0.05510 to 0.10657
```

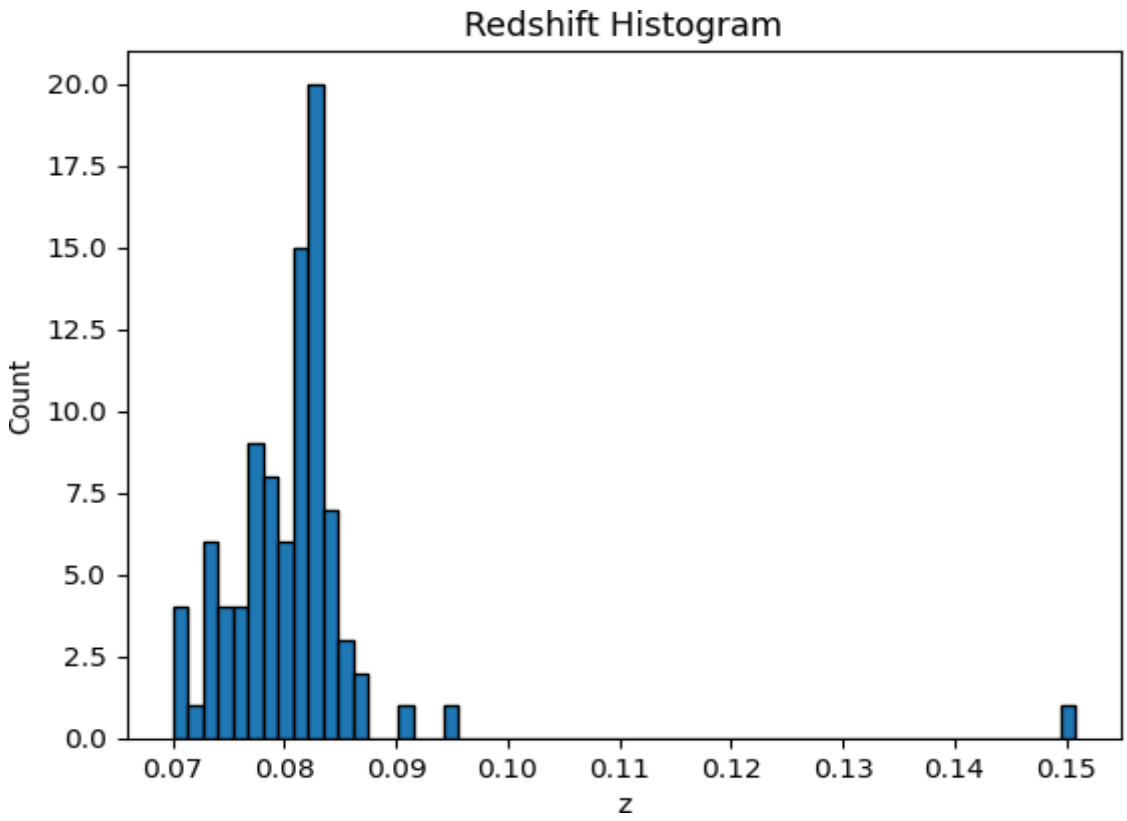
You can also use boxplot to visualize the overall values of redshift

```
# visualize the redshift distribution
plt.figure();
plt.boxplot(averaged_df["specz"], vert=False)
plt.title("Redshift Boxplot");
plt.xlabel("z");
plt.show()
```



But the best plot would be a histogram to see where most of the objects downloaded lie in terms of redshift value

```
plt.figure();
plt.hist(averaged_df["specz"], bins=60, edgecolor="black")
plt.title("Redshift Histogram");
plt.xlabel("z");
plt.ylabel("Count");
plt.show();
```



Filter your data based on the 3-sigma limit of redshift. You should remove all data points which are 3-sigma away from mean of redshift

```
# Filtering the data based on specz values, used 3 sigma deviation from mean as upper limi
cluster_df = averaged_df[(averaged_df["specz"] >= z_min) & (averaged_df["specz"] <= z_max)
print(f"Found {len(cluster_df)} cluster members out of {len(averaged_df)}")
```

```
Found 91 cluster members out of 92
```

Use the relation between redshift and velocity to add a column named velocity in the data. This would tell the expansion velocity at that redshift

```
cluster_df["velocity"] = (cluster_df["specz"].values * c_kms).to(u.km/u.s)
```

```
/tmp/ipython-input-13-869131744.py:1:  SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame. Try
using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  cluster_df["velocity"] = (cluster_df["specz"].values * c_kms).to(u.km/u.s)
```

use the dispersion equation to find something called velocity dispersion. You can even refer to wikipedia to know about the term wiki link here

It is the velocity dispersion value which tells us, some galaxies might be part of even larger groups!!

```
sigma_v_approx = np.std([v.value for v in cluster_df["velocity"]], ddof=1) * u.km/u.s
print("Approximate σᵥ =", sigma_v_approx)
```

```
Approximate σᵥ = 1316.152774680483 km / s
```

Step 2: Calculate Mean Redshift of the Cluster

We calculate the average redshift (`specz`) of galaxies that belong to a cluster. This gives us an estimate of the cluster's systemic redshift.

```
cluster_redshift = filtered_df['specz'].mean()
```

The velocity dispersion (v) of galaxies relative to the cluster mean redshift is computed using the relativistic Doppler formula:

$$v = c \cdot \frac{(1+z)^2 - (1+z_{\text{cluster}})^2}{(1+z)^2 + (1+z_{\text{cluster}})^2}$$

where:

- (v) is the relative velocity (dispersion),
- (z) is the redshift of the individual galaxy,
- (z_{cluster}) is the mean cluster redshift,
- (c) is the speed of light.

```
filtered_df = averaged_df[(averaged_df["specz"] >= z_min) & (averaged_df["specz"] <= z_max
z_c    = filtered_df["specz"].mean()
z_arr = filtered_df["specz"].values
```

```
v_rel = (c_kms * ((1+z_arr)**2 - (1+z_c)**2) / ((1+z_arr)**2 + (1+z_c)**2)).to(u.km/u.s)
filtered_df["velocity_rel"] = v_rel
disp = np.std(v_rel.value, ddof=1) * u.km/u.s
```

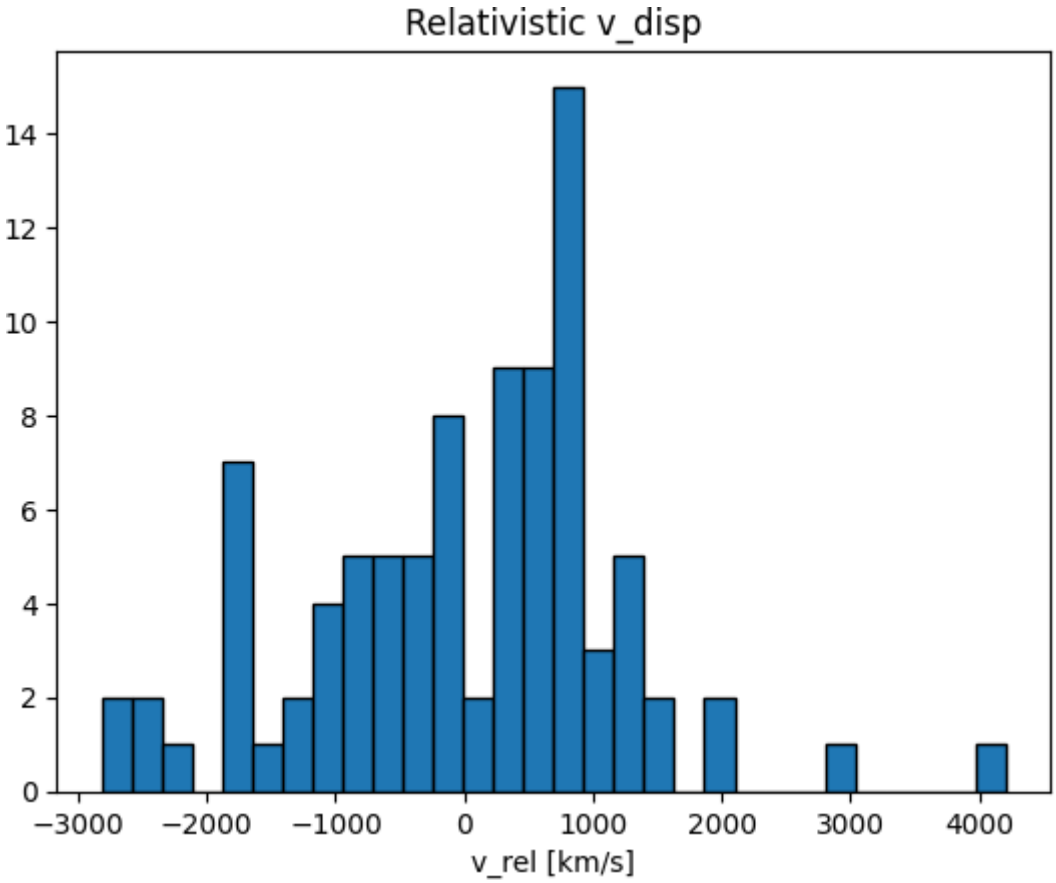
```
/tmp/ipython-input-17-3673429257.py:2:  SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  filtered_df["velocity_rel"] = v_rel
```

```
print(f"Cluster mean redshift = {z_c:.5f}")
print("Velocity dispersion σᵥ =", disp)
```

```
Cluster mean redshift = 0.08007
Velocity dispersion σᵥ = 1218.4929446822505 km / s
```

```
plt.figure();
plt.hist(v_rel.value, bins=30, edgecolor="black")
plt.title("Relativistic v_disp");
plt.xlabel("v_rel [km/s]");
plt.show();
```



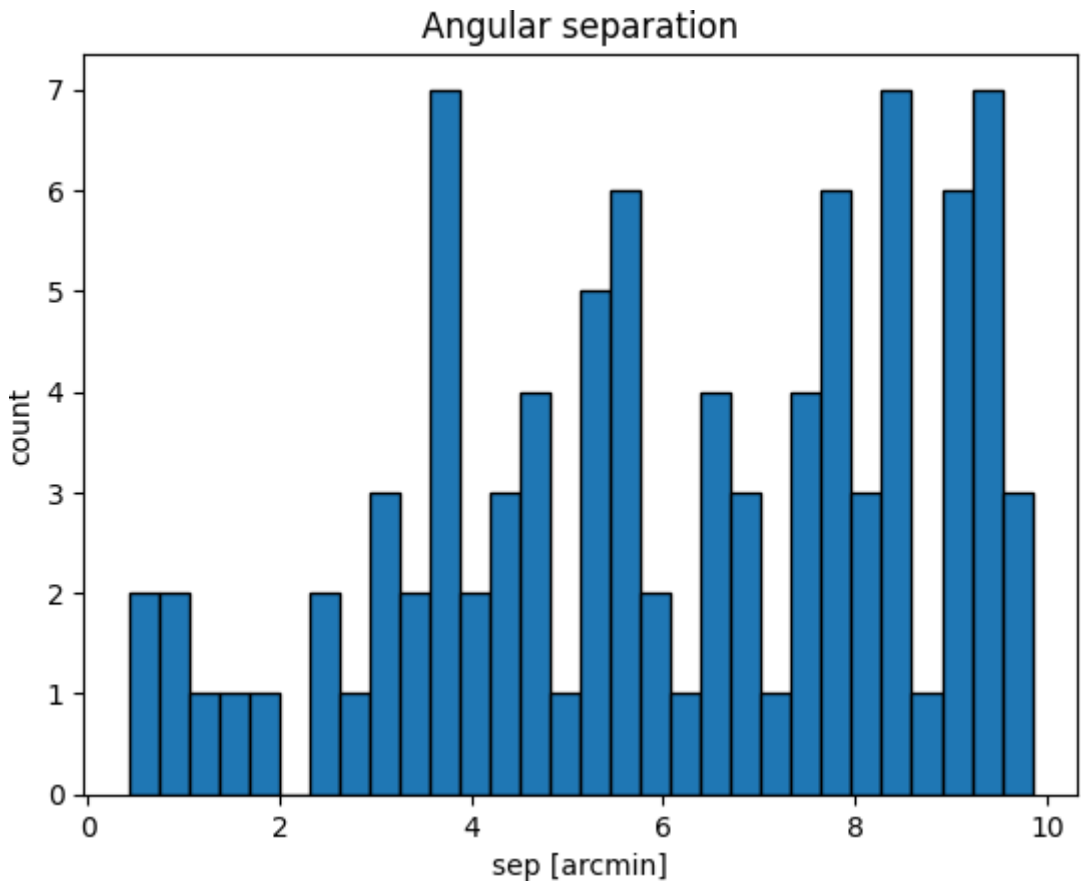
Pro tip: Check what the describe function of pandas does. Does it help to get quick look stats for your column of dispersion??

Step 4: Visualizing Angular Separation of Galaxies

We plot a histogram of the projected (angular) separation of galaxies from the cluster center. This helps us understand the spatial distribution of galaxies within the cluster field.

- The x-axis represents the angular separation (in arcminutes or degrees, depending on units).
- The y-axis shows the number of galaxies at each separation bin.

```
plt.figure();
plt.hist(filtered_df["proj_sep"], bins=30, edgecolor="black");
plt.title("Angular separation");
plt.xlabel("sep [arcmin]");
plt.ylabel("count");
plt.show()
```



Determining size and mass of the cluster:

Step 5: Estimating Physical Diameter of the Cluster

We now estimate the **physical diameter** of the galaxy cluster using cosmological parameters.

- `r` is the **co-moving distance**, approximated using a Taylor expansion for low redshift:

$$r = \frac{cz}{H_0}\left(1 - \frac{z}{2}(1 + q_0)\right)$$

  where q_0 is the deceleration parameter
- `ra` is the **angular diameter distance**, given by:

$$D_A = \frac{r}{1+z}$$

- Finally, we convert the observed angular diameter (in arcminutes) into physical size using:

$$\text{diameter (in Mpc)} = D_A \cdot \theta$$

  where θ is the angular size in radians, converted from arcminutes.

> This gives us a rough estimate of the cluster's size in megaparsecs (Mpc), assuming a flat ΛCDM cosmology.

```
theta       = np.median(filtered_df["proj_sep"]) * u.arcmin
theta_rad   = theta.to(u.rad)
```

```
r = (c_kms * z_c / H_0 * (1 - 0.5*z_c*(1+q0))).to(u.Mpc)  # low-z approximation
ra  = (r / (1+z_c)).to(u.Mpc) # angular-diameter
```

```
diameter = ra * theta_rad.value

print("Co-moving distance, r =", r)
print("Angular-diameter, ra  =", ra)
print("Cluster diameter      =", diameter)
```

```
Co-moving distance, r = 348.15152134598634 Mpc
Angular-diameter, ra  = 322.3422652698339 Mpc
Cluster diameter      = 0.593153720588005 Mpc
```

Step 6: Calculating the Dynamical Mass of the Cluster

We now estimate the **dynamical mass** of the galaxy cluster using the virial theorem:

$$M_{\text{dyn}} = \frac{3\sigma^2 R}{G}$$

Where:

- σ is the **velocity dispersion** in m/s (`disp * 1000`),
- R is the **cluster radius** in meters (half the physical diameter converted to meters),
- G is the **gravitational constant** in SI units,
- The factor of 3 assumes an isotropic velocity distribution (common in virial estimates).

We convert the final result into **solar masses** by dividing by 2×10^{30} kg.

> This mass estimate assumes the cluster is in dynamical equilibrium and bound by gravity.

```
R_m = (0.5 * diameter).to(u.m) # radius in meters
sigma_m = disp.to(u.m/u.s) # σᵥ in m/s
```

```
M_dyn = (3 * sigma_m**2 * R_m / G_si).to(u.M_sun)
```

```
D_L_pc = cosmo.luminosity_distance(filtered_df["specz"]).to(u.pc).value
m_r    = filtered_df["rmag"].values
M_r    = m_r - 5 * np.log10(D_L_pc/10)    # absolute mag
M_sun  = 4.64  # Sun's r-band M
L_r    = 10**(-0.4 * (M_r - M_sun))
M_L_ratio = 1.0    # assume 1 M⊙ per L⊙
luminous_mass = L_r * M_L_ratio
```

```
cluster_df["luminous_mass"] = luminous_mass
total_luminous_mass = luminous_mass.sum() * u.M_sun
```

```
/tmp/ipython-input-42-491636537.py:1:  SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame. Try
using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  cluster_df["luminous_mass"] = luminous_mass
```

```
print(f"Dynamical mass M_dyn     = {M_dyn.value:.2e} {M_dyn.unit}")
print(f"Total luminous mass      = {total_luminous_mass.value:.2e} {total_luminous_mass.uni
```

```
Dynamical mass M_dyn     = 3.07e+14 solMass
Total luminous mass      = 2.26e+12 solMass
```

Estimating Cosmological Parameters from Type Ia Supernovae

1. Introduction:

In this project, I explored how Type Ia supernovae can be used to determine some of the most important parameters in cosmology: the Hubble constant (H_0), the matter density parameter (Ω_m), and the age of the universe. Type Ia supernovae are considered "standard candles" because their intrinsic luminosity is well understood, allowing us to estimate their distances accurately. Using this property and observational data from the Pantheon+SH0ES dataset, I constructed and tested a flat ΛCDM cosmological model and evaluated how well it fits the data.

2. Methodology and Approach:

In this project, I employed a structured computational approach using Python. The required libraries included NumPy for numerical calculations, Pandas for data manipulation, Matplotlib for visualization, SciPy for statistical analysis and integration, and Astropy for handling astrophysical constants and units.

Initially, I downloaded and carefully cleaned the Pantheon+SH0ES dataset, which involved extracting relevant data columns—redshifts (zHD), distance moduli (MU_SH0ES), and their uncertainties (MU_SH0ES_ERR_DIAG). The cleaning process included converting columns to numeric types and removing any incomplete entries to maintain data integrity.

After preparing the data, I constructed the theoretical cosmological framework based on the flat ΛCDM model. This included defining the dimensionless Hubble parameter E(z), calculating the comoving distance through numerical integration using `quad`, and determining the luminosity distance $d_L(z) = (1 + z)D_C$. I then used this distance to calculate the theoretical distance modulus μ(z) through the relation $\mu = 5log_{10}(d_L/Mpc) + 25$.

To fit the model, I used `curve_fit` from SciPy, performing a non-linear least squares optimization to estimate the best-fit values of H_0 and Ω_m. I also tested an alternative fitting approach by fixing Ω_m at a range of values from 0.1 to 0.5 and re-fitting H_0, which allowed me to observe how both H_0 and the inferred age of the universe changed with different matter densities.

To verify the quality of the fit, I analyzed the residuals (μ_obs − μ_model) across all redshifts. I also performed segmented fitting by dividing the dataset into low-redshift ($z < 0.1$) and high-redshift ($z \geq 0.1$) subsets. For each subset, I repeated the fixed-Ω_m fits and computed the corresponding H_0 and age of the universe. These comparisons helped me understand how redshift influences parameter estimates and allowed for a more nuanced interpretation of cosmological trends.

Finally, I visualized the Hubble diagram and the residuals to assess how well the model fit the data. I also created tables summarizing the fitted H_0 and universe age for various fixed Ω_m values, both for the full dataset and for the low/high redshift subsets, revealing important trends and dependencies.

3. My Responses:

a) What value of the Hubble constant ($H0$) did you obtain from the full dataset?

I obtained $H_0 = 72.97 \pm 0.26$ km/s/Mpc when fitting the entire Pantheon+SH0ES dataset. This value was derived by simultaneously fitting for both H_0 and Ω_m using the theoretical model described above.

b) How does your estimated $H0$ compare with the Planck18 measurement of the same?

The Planck18 result gives $H_0 \approx 67.4$ km/s/Mpc, which is significantly lower than my result. This difference is part of the well-known "Hubble tension," which refers to the disagreement between early-universe measurements (like Planck CMB data) and late-universe measurements (like supernovae and Cepheids). My result is closer to the SH0ES measurement, which supports a higher value for H_0.

c) **What is the age of the Universe based on your value of H_0? (Assume $\Omega m = 0.3$) How does it change for different values of Ωm?**

Using $H_0 = 72.97$ km/s/Mpc and $\Omega_m = 0.351$ (the best-fit value), I calculated the age of the universe to be approximately 12.36 Gyr. I then tested several fixed Ω_m values ranging from 0.1 to 0.5. For low Ω_m values (e.g., 0.1), the universe appears older (≈16.44 Gyr), and for higher Ω_m values (e.g., 0.5), it appears younger (≈11.37 Gyr).

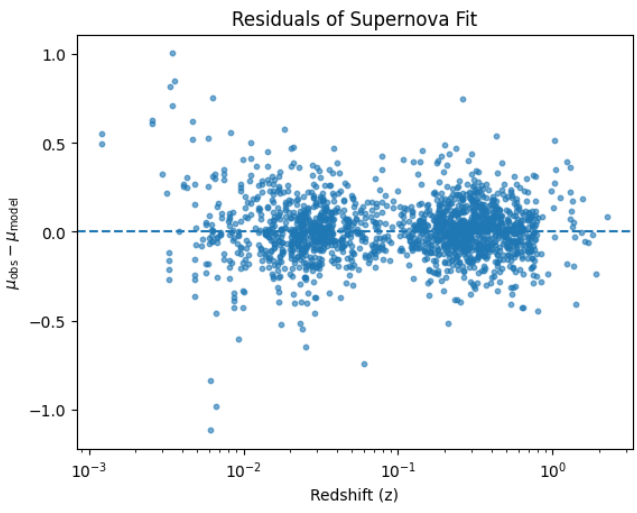
```
Ωm = 0.10 -> H0 = 76.03 ± 0.18 km/s/Mpc, Age = 16.44 Gyr
Ωm = 0.15 -> H0 = 75.35 ± 0.18 km/s/Mpc, Age = 15.03 Gyr
Ωm = 0.20 -> H0 = 74.71 ± 0.17 km/s/Mpc, Age = 14.08 Gyr
Ωm = 0.25 -> H0 = 74.10 ± 0.17 km/s/Mpc, Age = 13.38 Gyr
Ωm = 0.30 -> H0 = 73.53 ± 0.17 km/s/Mpc, Age = 12.82 Gyr
Ωm = 0.35 -> H0 = 72.98 ± 0.17 km/s/Mpc, Age = 12.37 Gyr
Ωm = 0.40 -> H0 = 72.46 ± 0.17 km/s/Mpc, Age = 11.98 Gyr
Ωm = 0.45 -> H0 = 71.96 ± 0.17 km/s/Mpc, Age = 11.65 Gyr
Ωm = 0.50 -> H0 = 71.48 ± 0.17 km/s/Mpc, Age = 11.37 Gyr
```

This shows that the age of the universe is quite sensitive to the matter density parameter. The image from my python notebook showing how the values of H_0 and age of universe are changing by varying the value of Ω_m.

d) Discuss the difference in *H*0 values obtained from the low-z and high-z samples. What could this imply?

When I split the dataset into low-z ($z < 0.1$) and high-z ($z \geq 0.1$), I found $H_0 = 73.01 \pm 0.28$ km/s/Mpc for low-z and $H_0 = 73.85 \pm 0.22$ km/s/Mpc for high-z. Although the difference is within error bars, it may suggest subtle redshift-dependent effects or evolution in the supernova sample. It also reflects how measurements at different epochs or scales can influence the inferred value of H_0.

e) Plot the residuals and comment on any trends or anomalies you observe.



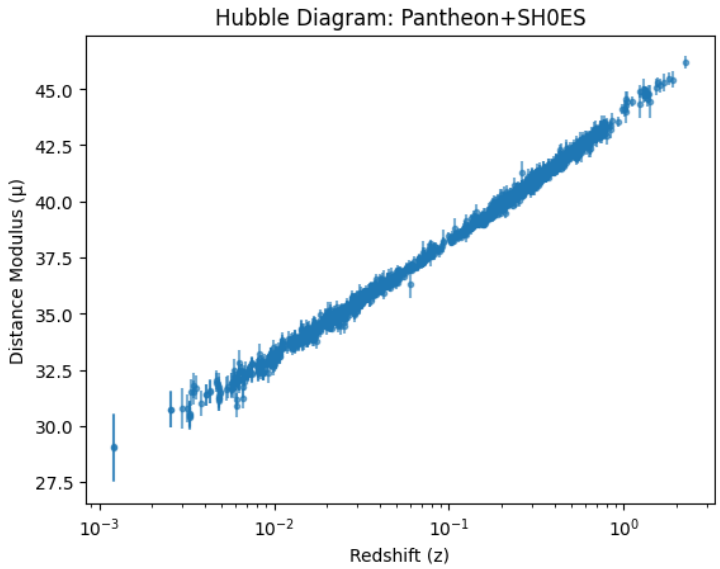
I plotted the residuals (μ_obs − μ_model) against redshift on a logarithmic scale. The scatter plot showed that the residuals are evenly distributed around zero across the entire redshift range, with no significant trends or deviations. There was slightly more scatter at lower redshifts, which is expected due to observational uncertainties and peculiar velocities, but overall the distribution appeared symmetric and unbiased. This supports the conclusion that the flat ΛCDM model is a good fit for the data. The absence of systematic trends in the residuals indicates that the theoretical model accurately captures the observed redshift-distance relationship.

f) What assumptions were made in the cosmological model, and how might relaxing them affect your results?

In my analysis, I assumed the standard flat ΛCDM model. This includes several key assumptions: (1) the universe is spatially flat ($\Omega_k = 0$), (2) dark energy is represented as a cosmological constant (with equation-of-state parameter $w = -1$), and (3) the matter content of the universe is composed of cold dark matter and baryons, with no contribution from radiation or curvature at late times. These assumptions simplify the equations and reduce the number of free parameters, which helps in fitting the data robustly. However, they may also mask underlying complexities. For example, allowing Ω_k to vary (i.e., a non-flat universe) could shift the inferred values of H_0 and Ω_m. Similarly, if dark energy is not a cosmological constant but evolves with time ($w \neq -1$), this could also alter the shape of the redshift-distance relation and change the best-fit parameters. Additionally, alternative cosmologies—such as dynamical dark energy models, modified gravity theories, or models with early dark energy—may better reconcile the current Hubble tension. By relaxing the flatness or fixed-w assumptions, we open up the parameter space, which could potentially lead to a model that fits both early- and late-universe data more consistently.

g) Based on the redshift-distance relation, what can we infer about the expansion history of the Universe?

The redshift-distance relation as visualized in the Hubble diagram clearly shows a logarithmic and monotonic increase in distance modulus (μ) with increasing redshift (z), forming a smooth upward curve. This indicates that supernovae farther away appear dimmer not only because they are more distant but also due to the accelerated expansion of the universe. The gentle upward curvature confirms that at lower redshifts the expansion was slower, and as we move to higher redshifts,



the expansion becomes more pronounced. This is strong evidence for an accelerating universe, which aligns with the presence of dark energy. The shape of this curve validates the ΛCDM model and implies that the universe's expansion has changed over time — it was decelerating in the past due to matter domination and has been accelerating recently due to dark energy.

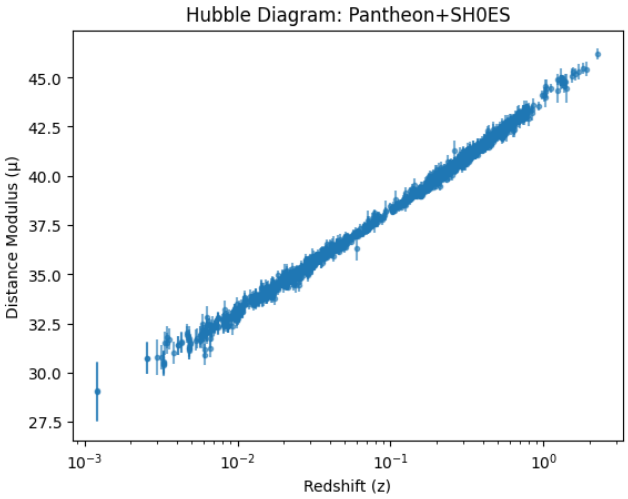
4. Observations and Interpretations:

Based on the results from all stages of the analysis, I made several key observations:

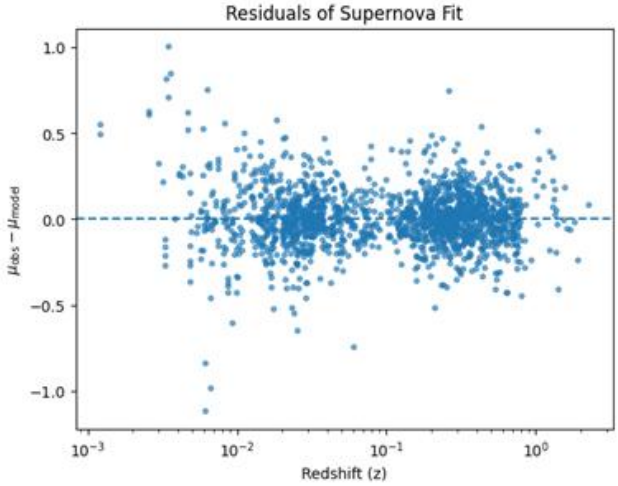
- The best-fit values of H_0 from the full dataset, the fixed Ω_m analysis, and the redshift-split subsets are all in excellent agreement with recent late-universe measurements (e.g., SH0ES), but remain significantly higher than the Planck18 result. This reinforces the ongoing Hubble tension in cosmology.
- The age of the universe showed a clear inverse relationship with Ω_m. As Ω_m increased, the calculated age decreased, ranging from about 16.4 Gyr (for $\Omega_m = 0.1$) to about 11.4 Gyr (for $\Omega_m = 0.5$). This was consistent both in the full dataset analysis and in the redshift-split subsets. This trend highlights the sensitivity of cosmological conclusions to the assumed matter density.
- The values of H_0 derived separately from low-z and high-z subsets showed slight but systematic differences. For example, at $\Omega_m = 0.3$, H_0_low = 73.01 km/s/Mpc and H_0_high = 73.85 km/s/Mpc. This could hint at redshift evolution or population effects in the supernova sample, or subtle systematic biases.
- The residuals plot showed excellent agreement between the theoretical model and observational data. The scatter was symmetrical and centered around zero, especially for mid- and high-redshift regions. The increased spread at very low redshifts is expected due to local peculiar velocities and observational noise.

These findings together confirm the validity of the flat ΛCDM model while also demonstrating that even within such a successful model, small variations in input assumptions or sample segmentation can yield slightly different outcomes.

5. Graphical Visualization:



I plotted distance modulus (μ) against redshift (z) on a logarithmic x-axis. The plot shows a clear increasing trend consistent with cosmic expansion and matched well with the theoretical model.



The residuals between observed and modeled distance moduli were plotted against redshift. These were well-centered around zero and showed no strong trends, confirming that the model fits the data robustly.

Ω_m = 0.10	H_0 = 76.03 ± 0.18 km/s/Mpc	Age = 16.44 Gyr
Ω_m = 0.15	H_0 = 75.35 ± 0.18 km/s/Mpc	Age = 15.03 Gyr
Ω_m = 0.20	H_0 = 74.71 ± 0.17 km/s/Mpc	Age = 14.08 Gyr
Ω_m = 0.25	H_0 = 74.10 ± 0.17 km/s/Mpc	Age = 13.38 Gyr
Ω_m = 0.30	H_0 = 73.53 ± 0.17 km/s/Mpc	Age = 12.82 Gyr
Ω_m = 0.35	H_0 = 72.98 ± 0.17 km/s/Mpc	Age = 12.37 Gyr
Ω_m = 0.40	H_0 = 72.46 ± 0.17 km/s/Mpc	Age = 11.98 Gyr
Ω_m = 0.45	H_0 = 71.96 ± 0.17 km/s/Mpc	Age = 11.65 Gyr
Ω_m = 0.50	H_0 = 71.48 ± 0.17 km/s/Mpc	Age = 11.37 Gyr

This table shows how H_0 and the age of the universe change as Ω_m is varied from 0.1 to 0.5 using the full dataset. It visually demonstrates how increasing Ω_m leads to a younger universe and slightly lower H_0 values.

Ω_m	H_0_low ± err	Age_low (Gyr)	H_0_high ± err	Age_high (Gyr)
0.10	73.39 ± 0.28	17.03	77.67 ± 0.23	16.09
0.15	73.29 ± 0.28	15.45	76.62 ± 0.23	14.78
0.20	73.20 ± 0.28	14.37	75.64 ± 0.22	13.91
0.25	73.10 ± 0.28	13.56	74.72 ± 0.22	13.27
0.30	73.01 ± 0.28	12.91	73.85 ± 0.22	12.77
0.35	72.91 ± 0.28	12.38	73.03 ± 0.22	12.36
0.40	72.82 ± 0.28	11.92	72.24 ± 0.21	12.02
0.45	72.73 ± 0.28	11.53	71.50 ± 0.21	11.73
0.50	72.63 ± 0.28	11.19	70.79 ± 0.21	11.48

This second table compares H_0 and the age of the universe for low-z and high-z subsets across the same Ω_m range. It highlights how the two subsets yield slightly different values and reinforces the redshift-dependence of the results.

6. Conclusion:

This project allowed me to engage deeply with real astronomical data and apply fundamental cosmological models to interpret it. Through the step-by-step modeling and analysis of the Pantheon+SH0ES supernova dataset, I was able to estimate the Hubble constant, explore the effects of the matter density parameter, and compute the corresponding age of the universe. I also examined how model assumptions and data segmentation influence the results, which gave me a more critical understanding of both the power and limitations of cosmological inference.

Overall, the project not only solidified my technical skills in data analysis and scientific computing but also highlighted the nuances behind contemporary debates like the Hubble tension. It was interesting to see how small changes in assumptions or methodology can lead to differences in conclusions. This hands-on experience has strengthened my appreciation for the interplay between theoretical physics and observational data in modern astrophysics.

7. Python Notebook (Appendix)

🔭 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 [ ]:
```
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

In [ ]:
```
# Path to the Pantheon+SH0ES data file
file_path = 'Pantheon+SH0ES.dat'

# Load the file
df = pd.read_csv(file_path, delim_whitespace=True, comment='#', header=None)
```

```
/tmp/ipython-input-4-3529426103.py:5: FutureWarning: The 'delim_whitespace' keyword in pd.re
ad_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='#', header=None)
```

In [ ]:
```
df.head()
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	
0	CID	IDSURVEY	zHD	zHDERR	zCMB	zCMBERR	zHEL	zHELERR	m_b_corr	m_b_corr_err
1	2011fe	51	0.00122	0.00084	0.00122	2e-05	0.00082	2e-05	9.74571	1.
2	2011fe	56	0.00122	0.00084	0.00122	2e-05	0.00082	2e-05	9.80286	1.
3	2012cg	51	0.00256	0.00084	0.00256	2e-05	0.00144	2e-05	11.4703	0.7
4	2012cg	56	0.00256	0.00084	0.00256	2e-05	0.00144	2e-05	11.4919	0.7

5 rows × 47 columns

🗒️ 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 [ ]:
```
df.columns = df.iloc[0]
```

In [ ]:
```
df = df.drop(index=0).reset_index(drop=True)
```

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['CID', 'IDSURVEY', 'zHD', 'zHDERR', 'zCMB', 'zCMBERR', 'zHEL',
       'zHELERR', 'm_b_corr', 'm_b_corr_err_DIAG', 'MU_SH0ES',
       'MU_SH0ES_ERR_DIAG', 'CEPH_DIST', 'IS_CALIBRATOR', 'USED_IN_SH0ES_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'],
      dtype='object', name=0)
```

🧹 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 [ ]:
```
# Convert relevant columns to numeric, coercing invalid entries to NaN
df['zHD'] = pd.to_numeric(df['zHD'], errors='coerce')
df['MU_SH0ES'] = pd.to_numeric(df['MU_SH0ES'], errors='coerce')
df['MU_SH0ES_ERR_DIAG'] = pd.to_numeric(df['MU_SH0ES_ERR_DIAG'], errors='coerce')
```

In [ ]:
```
# Drop any rows missing values in our key columns
df_clean = df.dropna(subset=['zHD', 'MU_SH0ES', 'MU_SH0ES_ERR_DIAG'])
```

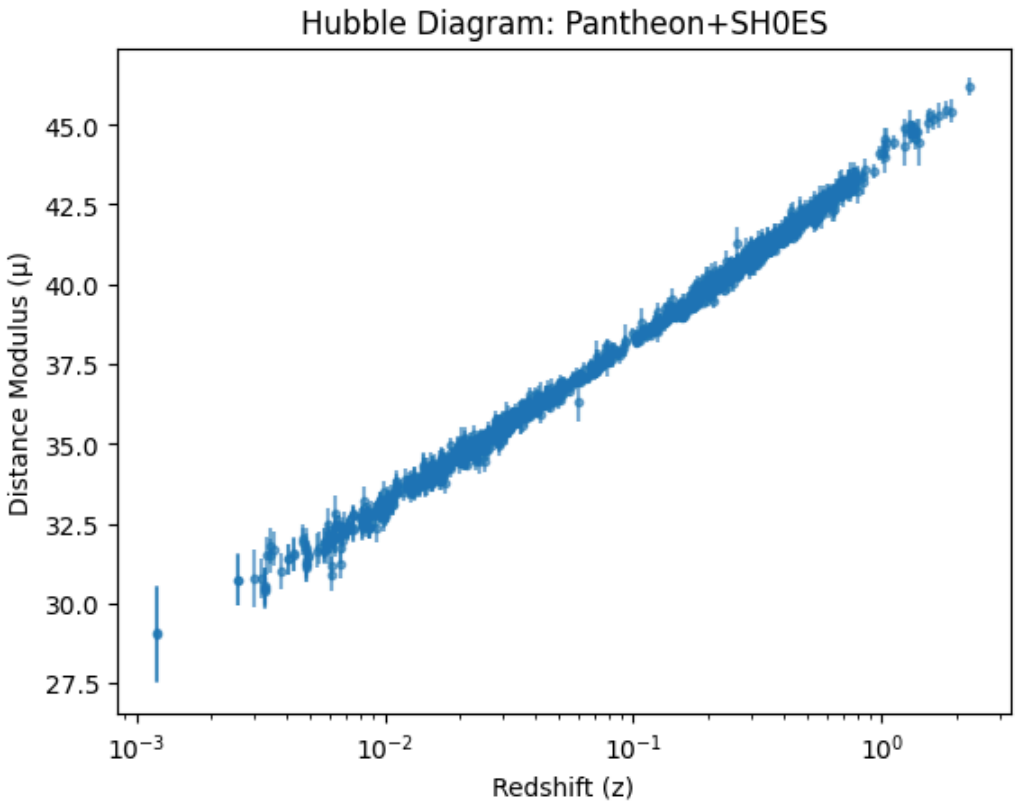
In [ ]:
```
# Extract NumPy arrays for modeling
z = df_clean['zHD'].values
mu = df_clean['MU_SH0ES'].values
mu_err = df_clean['MU_SH0ES_ERR_DIAG'].values
```

📈 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 [ ]:
```
plt.figure()
plt.errorbar(z, mu, yerr=mu_err, fmt='.', alpha=0.6)
plt.xscale('log')
plt.xlabel('Redshift (z)')
plt.ylabel('Distance Modulus (μ)')
plt.title('Hubble Diagram: Pantheon+SH0ES')
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 [ ]:
```python
# Define the E(z) for flat LCDM
# Dimensionless Hubble parameter for flat ΛCDM
def E(z, Omega_m):
    return np.sqrt(Omega_m * (1 + z)**3 + (1.0 - Omega_m))
```

In [ ]:
```python
# Luminosity distance in Mpc, try using scipy quad to integrate.
def luminosity_distance(z, H0, Omega_m):

    if not hasattr(H0, 'unit'):   # Ensure H0 is a Quantity with units km/s/Mpc
        H0 = H0 * (u.km / u.s / u.Mpc)

    c_km_s = c.to(u.km / u.s)  # Speed of light in km/s
    factor = (c_km_s / H0).to(u.Mpc)  # Prefactor c/H0 has units of Mpc

    def integrand(zp):
        return 1.0 / E(zp, Omega_m)

    # Integrate for comoving distance
    if np.isscalar(z):
        Dc = factor * quad(integrand, 0, z)[0]
    else:
        Dc = np.array([quad(integrand, 0, zi)[0] for zi in z]) * factor

    D_L = (1 + z) * Dc # Luminosity distance D_L = (1+z) * D_C
    return D_L.to(u.Mpc)
```

In [ ]:
```python
# Theoretical distance modulus
def mu_theory(z, H0, Omega_m):
    D_L = luminosity_distance(z, H0, Omega_m)
    return 5 * np.log10(D_L.value) + 25   # Convert to dimensionless for log10
```

🔧 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 [ ]:
```
# Initial guess: H0 = 70 km/s/Mpc, Omega_m = 0.3
p0 = [70, 0.3]
```

In [ ]:
```
# Fit mu_theory(z, H0, Omega_m) to the data
popt, pcov = curve_fit(
    mu_theory,
    z, mu,
    sigma=mu_err,
    p0=p0,
    absolute_sigma=True,
    maxfev=5000
)
```

In [ ]:
```
# Extract best-fit parameters and their 1σ uncertainties
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
```

⌛ 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 [ ]:
```
def age_of_universe(H0, Omega_m):
    H0_si = (H0 * u.km / u.s / u.Mpc).to(1 / u.s)  # Convert H0 [km/s/Mpc] to s⁻¹
    integral = quad(lambda z: 1.0 / ((1 + z) * E(z, Omega_m)), 0, np.inf, limit=200)[0]
    t0_sec = integral / H0_si.value  # Age in seconds = integral / H0_si
    return (t0_sec * u.s).to(u.Gyr).value # Convert to gigayears
```

In [ ]:
```
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

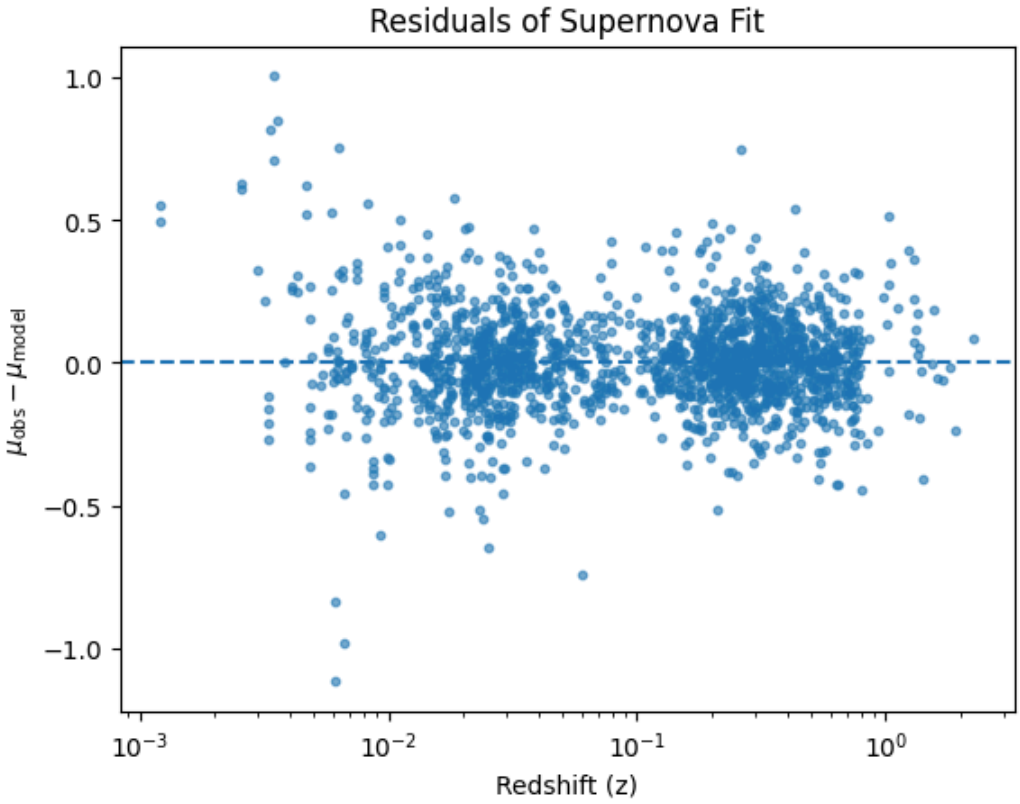
To evaluate how well our cosmological model fits the data, we compute the residuals:

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

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 [ ]:
```
mu_pred = mu_theory(z, H0_fit * u.km/u.s/u.Mpc, Omega_m_fit)
residuals = mu - mu_pred
```

In [ ]:
```
# Plot residuals vs. redshift
plt.figure()
plt.scatter(z, residuals, s=10, alpha=0.6)
plt.xscale('log')
plt.axhline(0, linestyle='--')
plt.xlabel('Redshift (z)')
plt.ylabel(r'$\mu_{\rm obs} - \mu_{\rm model}$')
plt.title('Residuals of Supernova Fit')
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 [ ]:
```
# Fix Omega_m = 0.3, fit only H0
def mu_fixed_Om(z, H0):
    return mu_theory(z, H0, Omega_m=0.3)
```

In [ ]:
```
# Perform the fit
popt, pcov = curve_fit(
    mu_fixed_Om,
    z, mu,
    sigma=mu_err,
    p0=[70],
    absolute_sigma=True,
    maxfev=5000
)
```

In [ ]:
```
# Extract result
H0_fixed, = popt
H0_fixed_err, = np.sqrt(np.diag(pcov))
```

```
print(f"Fixed Ωm=0.3 → H0 = {H0_fixed:.2f} ± {H0_fixed_err:.2f} km/s/Mpc")
```

```
Fixed Ωm=0.3 → H0 = 73.53 ± 0.17 km/s/Mpc
```

Checking with other fixed values of Ω_m

In [ ]:
```
omega_values = [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5]
results = []
```

In [ ]:
```
for Om in omega_values:
    # Define model with Ωₘ fixed
    def mu_fixed(z, H0):
        return mu_theory(z, H0, Omega_m=Om)

    # Fit for H0
    popt, pcov = curve_fit(
        mu_fixed,
        z, mu,
        sigma=mu_err,
        p0=[70],
        absolute_sigma=True,
        maxfev=5000
    )
    H0_fit = popt[0]
    H0_err = np.sqrt(pcov[0,0])

    # Compute age of universe for this (H0, Om)
    t0 = age_of_universe(H0_fit, Om)

    results.append((Om, H0_fit, H0_err, t0))
    print(f"Ωₘ = {Om:.2f} -> H₀ = {H0_fit:.2f} ± {H0_err:.2f} km/s/Mpc, Age = {t0:.2f} Gyr")
```

```
Ωₘ = 0.10 -> H₀ = 76.03 ± 0.18 km/s/Mpc, Age = 16.44 Gyr
Ωₘ = 0.15 -> H₀ = 75.35 ± 0.18 km/s/Mpc, Age = 15.03 Gyr
Ωₘ = 0.20 -> H₀ = 74.71 ± 0.17 km/s/Mpc, Age = 14.08 Gyr
Ωₘ = 0.25 -> H₀ = 74.10 ± 0.17 km/s/Mpc, Age = 13.38 Gyr
Ωₘ = 0.30 -> H₀ = 73.53 ± 0.17 km/s/Mpc, Age = 12.82 Gyr
Ωₘ = 0.35 -> H₀ = 72.98 ± 0.17 km/s/Mpc, Age = 12.37 Gyr
Ωₘ = 0.40 -> H₀ = 72.46 ± 0.17 km/s/Mpc, Age = 11.98 Gyr
Ωₘ = 0.45 -> H₀ = 71.96 ± 0.17 km/s/Mpc, Age = 11.65 Gyr
Ωₘ = 0.50 -> H₀ = 71.48 ± 0.17 km/s/Mpc, Age = 11.37 Gyr
```

🔍 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 [ ]:
```
# Redshift split threshold
z_split = 0.1

# Masks for low-z and high-z samples
mask_low  = z <  z_split
mask_high = z >= z_split
```

In [ ]:
```
# Fit low-z subsample
popt_low, pcov_low = curve_fit(
    mu_fixed_Om,
```

```
    z[mask_low],
    mu[mask_low],
    sigma=mu_err[mask_low],
    p0=[70],
    absolute_sigma=True,
    maxfev=5000
)

H0_low     = popt_low[0]
H0_low_err = np.sqrt(pcov_low[0,0])
```

In [ ]:
```
# Fit high-z subsample
popt_high, pcov_high = curve_fit(
    mu_fixed_Om,
    z[mask_high],
    mu[mask_high],
    sigma=mu_err[mask_high],
    p0=[70],
    absolute_sigma=True,
    maxfev=5000
)

H0_high     = popt_high[0]
H0_high_err = np.sqrt(pcov_high[0,0])
```

In [32]:
```
t0_low  = age_of_universe(H0_low,  0.3)
t0_high = age_of_universe(H0_high, 0.3)

print(f"Low-z sample → H₀ = {H0_low:.2f} km/s/Mpc, Age = {t0_low:.2f} Gyr")
print(f"High-z sample → H₀ = {H0_high:.2f} km/s/Mpc, Age = {t0_high:.2f} Gyr")
```

```
Low-z sample → H₀ = 73.01 km/s/Mpc, Age = 12.91 Gyr
High-z sample → H₀ = 73.85 km/s/Mpc, Age = 12.77 Gyr
```

In [33]:
```
omega_values = [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5]
z_split = 0.1
mask_low  = z <  z_split
mask_high = z >= z_split
```

In [41]:
```
print("Ωₘ   |  H₀_low ± err  | Age_low (Gyr) |  H₀_high ± err | Age_high (Gyr)")
print("-----|----------------|---------------|----------------|--------------")

for Om in omega_values:
    # Fit H₀ for low-z with Ωₘ fixed
    def mu_low(z_vals, H0):
        return mu_theory(z_vals, H0, Omega_m=Om)
    p_low, cov_low = curve_fit(mu_low,
                               z[mask_low], mu[mask_low],
                               sigma=mu_err[mask_low],
                               p0=[70],
                               absolute_sigma=True,
                               maxfev=5000)
    H0_low, H0_low_err = p_low[0], np.sqrt(cov_low[0,0])
    t0_low = age_of_universe(H0_low, Om)

    # Fit H₀ for high-z with Ωₘ fixed
    def mu_high(z_vals, H0):
        return mu_theory(z_vals, H0, Omega_m=Om)
    p_high, cov_high = curve_fit(mu_high,
                                 z[mask_high], mu[mask_high],
                                 sigma=mu_err[mask_high],
                                 p0=[70],
                                 absolute_sigma=True,
                                 maxfev=5000)
    H0_high, H0_high_err = p_high[0], np.sqrt(cov_high[0,0])
```

```
    t0_high = age_of_universe(H0_high, Om)
    print(f"{Om:>4.2f} | {H0_low:6.2f} ± {H0_low_err:<5.2f} | {t0_low:13.2f} | "f"{H0_high:
```

Ω_m	H_o_low ± err	Age_low (Gyr)	H_o_high ± err	Age_high (Gyr)
0.10	73.39 ± 0.28	17.03	77.67 ± 0.23	16.09
0.15	73.29 ± 0.28	15.45	76.62 ± 0.23	14.78
0.20	73.20 ± 0.28	14.37	75.64 ± 0.22	13.91
0.25	73.10 ± 0.28	13.56	74.72 ± 0.22	13.27
0.30	73.01 ± 0.28	12.91	73.85 ± 0.22	12.77
0.35	72.91 ± 0.28	12.38	73.03 ± 0.22	12.36
0.40	72.82 ± 0.28	11.92	72.24 ± 0.21	12.02
0.45	72.73 ± 0.28	11.53	71.50 ± 0.21	11.73
0.50	72.63 ± 0.28	11.19	70.79 ± 0.21	11.48

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