Summer School 2025

Astronomy & Astrophysics

Project Report

Prepared by

Student Name: ADITYA VISHWANATH

Institution Name: Vellore Institute of Technology, Chennai

Institution Roll No: 21BRS1384

ISA Admission No: 143677

Projects Name

1. **Estimating the Dynamical Mass of a Galaxy Cluster**
2. **Predicting the Hubble Parameter and the Age of the Universe using Supernovae Ia Data**

<u>Submitted To</u>

Name: Mr. Sahil Sakkarwal

Designation: Program Supervisor

Institution: Indian Space Academy

ASSIGNMENT 1

ESTIMATING THE DYNAMICAL MASS OF A GALAXY CLUSTER

dynamical-mass

July 1, 2025

#DYNAMICAL MASS PROJECT

0.0.1 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

> Tip: If you haven't used `pandas` before, it's worth learning as it offers powerful tools to manipulate and analyze structured datasets.

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

[2]:
```
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
from astropy.io import fits
```

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.

[3]:
```
# Constants:
#units also mentioned
from astropy import constants as const
from astropy import units as u

H0 = 70* u.km/u.s/u.Mpc # Hubble constant in SI unit that is in kilometre per␣
↪second per megaparsec
print(H0)
c_to_kms=const.c.to(u.km/u.s) # Speed of light in km/s
print(c_to_kms)
```

```
G=const.G.to(u.Mpc * u.km**2 / u.s**2 / u.Msun)  # Gravitational constant in␣
↪Mpc (km/s)^2 M_sun ¹
print(G)
q0= -0.534  # Deceleration parameter (assumed from Planck fit)
```

```
70.0 km / (Mpc s)
299792.458 km / s
4.300917270036279e-09 km2 Mpc / (solMass s2)
```

Read the csv data into the python using the method below

[6]:
```
df = pd.read_csv('Skyserver_SQL6_20_2025 5_07_17 PM.csv',header=1) # Data␣
↪downloaded from SDSS. header=1 as we start reading data from the 2nd row
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 139 entries, 0 to 138
Data columns (total 15 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   objid      139 non-null    float64
 1   ra         139 non-null    float64
 2   dec        139 non-null    float64
 3   photoz     139 non-null    float64
 4   photozerr  139 non-null    float64
 5   specz      139 non-null    float64
 6   speczerr   139 non-null    float64
 7   proj_sep   139 non-null    float64
 8   umag       139 non-null    float64
 9   umagerr    139 non-null    float64
 10  gmag       139 non-null    float64
 11  gmagerr    139 non-null    float64
 12  rmag       139 non-null    float64
 13  rmagerr    139 non-null    float64
 14  obj_type   139 non-null    int64  
dtypes: float64(14), int64(1)
memory usage: 16.4 KB
```

[7]:
```
df.head()
```

[7]:
```
          objid         ra        dec    photoz  photozerr     specz  \
0  1.240000e+18  257.82458  64.133257  0.079193   0.022867  0.082447   
1  1.240000e+18  257.82458  64.133257  0.079193   0.022867  0.082466   
2  1.240000e+18  257.83332  64.126043  0.091507   0.014511  0.081218   
3  1.240000e+18  257.85137  64.173247  0.081102   0.009898  0.079561   
4  1.240000e+18  257.85137  64.173247  0.081102   0.009898  0.079568   

   speczerr  proj_sep      umag  umagerr      gmag   gmagerr      rmag  \
```

```
0  0.000017  8.347733  18.96488  0.043377  17.49815  0.005672  16.75003
1  0.000014  8.347733  18.96488  0.043377  17.49815  0.005672  16.75003
2  0.000021  8.011259  20.22848  0.072019  18.38334  0.007763  17.46793
3  0.000022  8.739276  19.21829  0.050135  17.18970  0.004936  16.22043
4  0.000019  8.739276  19.21829  0.050135  17.18970  0.004936  16.22043

    rmagerr  obj_type
0  0.004708         3
1  0.004708         3
2  0.005828         3
3  0.003769         3
4  0.003769         3
```

[8]:
```python
for columns in df.columns:
  print(columns)
```

```
objid
ra
dec
photoz
photozerr
specz
speczerr
proj_sep
umag
umagerr
gmag
gmagerr
rmag
rmagerr
obj_type
```

0.0.2 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:

"'python 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()

[9]:
```python
# Calculating the average specz for each id:
averaged_df = df.groupby('objid').agg({'specz': 'mean','ra': 'first','dec':
↪'first','proj_sep': 'first',}).reset_index()
averaged_df #there is only 1 object id so thats why only 1 row
```

[9]:

```
          objid     specz         ra        dec  proj_sep
0  1.240000e+18  0.081047  257.82458  64.133257  8.347733
```

[10]:

```
averaged_df.describe()['specz']
#count,mean,std,min,25%,50%,75% values are taken from the above single row table
#count=1 as there is only 1 row.
```

[10]:

```
count    1.000000
mean     0.081047
std           NaN
min      0.081047
25%      0.081047
50%      0.081047
75%      0.081047
max      0.081047
Name: specz, 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

[11]:

```
import numpy as np

#Mean and std of the averaged specz values
mean_specz = df['specz'].mean()
std_specz = df['specz'].std()
# Computing the 3-sigma limits
lower_limit = mean_specz - 3 * std_specz      #Lower limit= -3
upper_limit = mean_specz + 3 * std_specz      #Upper limit= +3


print(f"Mean redshift: {mean_specz:.5f}")
print(f"Standard deviation: {std_specz:.5f}")
print(f"3-sigma lower limit: {lower_limit:.5f}")
print(f"3-sigma upper limit: {upper_limit:.5f}")
```

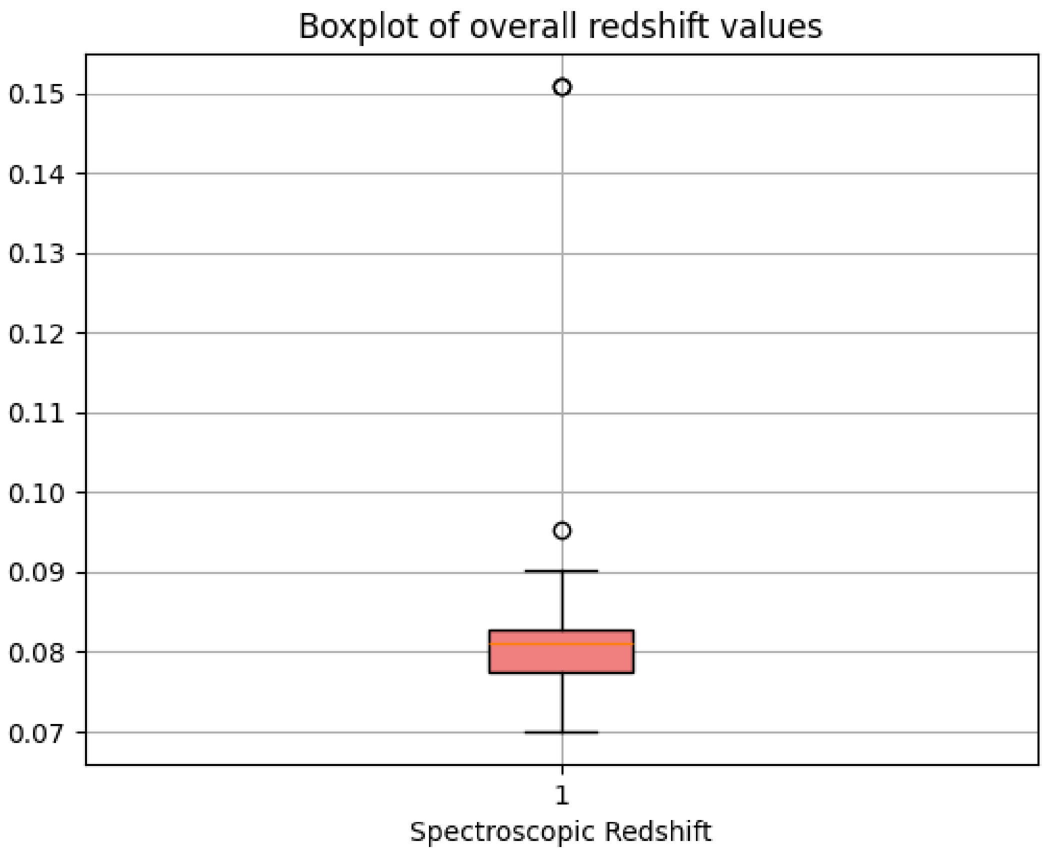
```
Mean redshift: 0.08105
Standard deviation: 0.00950
3-sigma lower limit: 0.05255
3-sigma upper limit: 0.10954
```

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

[12]:

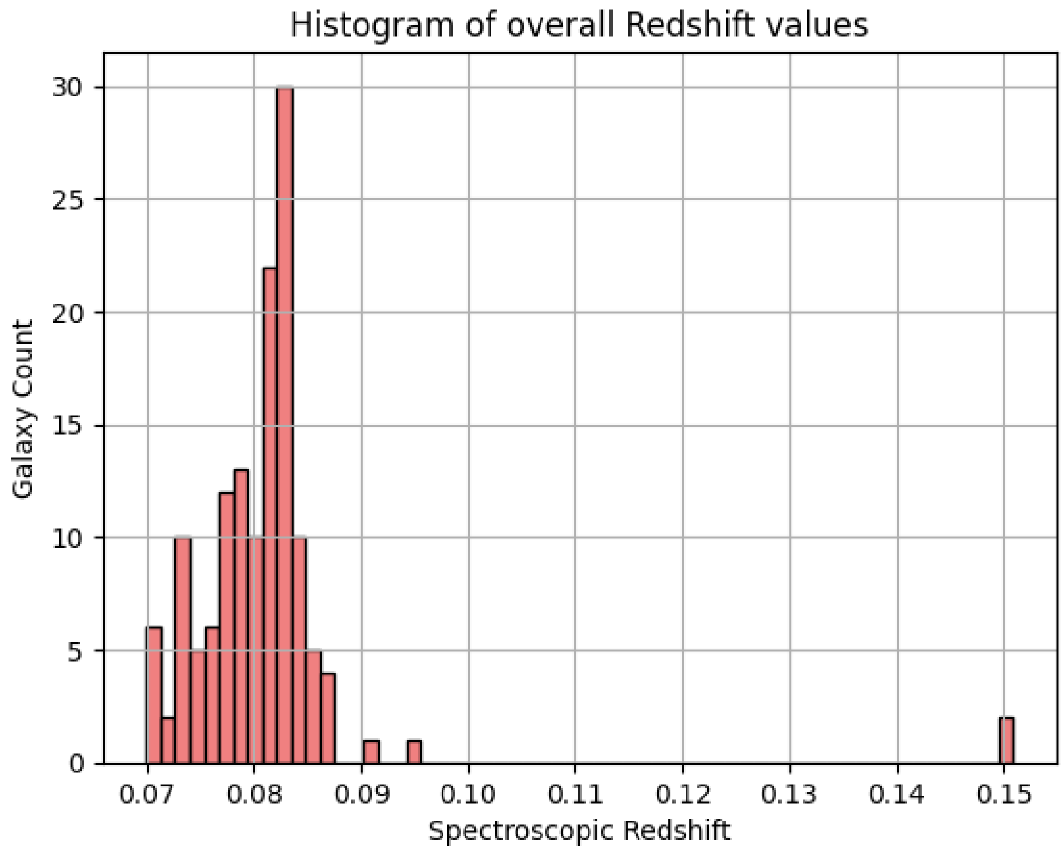
```
import matplotlib.pyplot as plt
# Boxplot of overall redshift value
plt.boxplot(df['specz'],␣
 ↪patch_artist=True,boxprops=dict(facecolor='lightcoral'))
plt.title('Boxplot of overall redshift values')
```

```
plt.xlabel('Spectroscopic Redshift')
plt.grid(True)
plt.show()
```



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

[13]:
```
# Histogram of redshifts
plt.hist(df['specz'], color='lightcoral', edgecolor='black',bins=60)
plt.xlabel('Spectroscopic Redshift')
plt.ylabel('Galaxy Count')
plt.title('Histogram of overall Redshift values')
plt.grid(True)
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

[14]:
```
# Filtering the data based on specz values, used 3 sigma deviation from mean as␣
↪upper limit.
filtered_df = df[(df['specz'] >= lower_limit) & (df['specz'] <= upper_limit)]
print("Original count:", len(df))
print("Filtered count:", len(filtered_df)) #only 2 values were not in the range␣
↪of upper limit and lower limit
```

```
Original count: 139
Filtered count: 137
```

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

[15]:
```
filtered_df['velocity'] = filtered_df['specz'] * c_to_kms
filtered_df['velocity'].head()
```

```
/tmp/ipython-input-15-1625550812.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

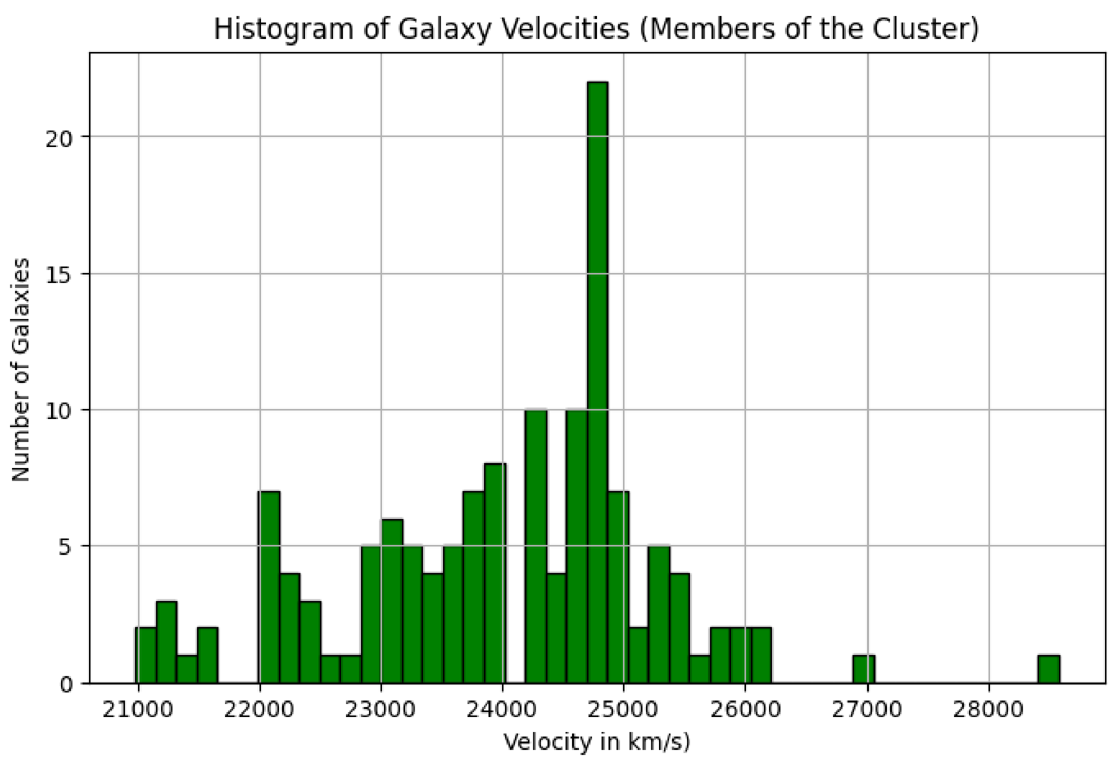
```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df['velocity'] = filtered_df['specz'] * c_to_kms
```

[15]:

```
0    24717.081720
1    24722.783773
2    24348.666769
3    23851.808736
4    23853.802356
Name: velocity, dtype: float64
```

[16]:

```python
#plot the velocity column created as hist
import matplotlib.pyplot as plt

# Plot histogram of galaxy velocities
plt.figure(figsize=(8, 5))
plt.hist(filtered_df['velocity'], bins=45, color='green', edgecolor='black')
plt.xlabel('Velocity in km/s)')
plt.ylabel('Number of Galaxies')
plt.title('Histogram of Galaxy Velocities (Members of the Cluster)')
plt.grid(True)
plt.show()
```



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!!

0.0.3 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.

[17]:
```
import numpy as np
from astropy.constants import c
import astropy.units as u

#Calculation of mean redshift of the cluster(z_cluster)
z_cluster = filtered_df['specz'].mean()
print(z_cluster)

z = filtered_df['specz'] #redshift of the individual galaxy

numerator = (1 + z)**2 - (1 + z_cluster)**2 #numerator of relativistic Doppler␣
↪formula
denominator = (1 + z)**2 + (1 + z_cluster)**2 # denominator of relativistic␣
↪Doppler formula

filtered_df['velocity_dispersion'] = c_to_kms * (numerator/denominator) #␣
↪velocity dispersion ( v ) of galaxies relative to the cluster mean redshift␣
↪is computed using the relativistic Doppler formula
filtered_df['velocity_dispersion'].head()
```

```
0.08002739656934307

/tmp/ipython-input-17-2395977292.py:14: 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_guide/indexing.html#returning-a-view-versus-a-copy
```

filtered_df['velocity_dispersion'] = c_to_kms * (numerator/denominator) #
velocity dispersion (v) of galaxies relative to the cluster mean redshift is
computed using the relativistic Doppler formula

[17]:
```
0     670.963582
1     676.231250
2     330.417516
3    -129.470206
4    -127.623517
Name: velocity_dispersion, dtype: float64
```

[18]:
```
sigma_v = filtered_df['velocity_dispersion'].std() #standard deviation of the␣
↪velocity dispersion of cluster
print(f"Velocity dispersion ( ): {sigma_v:.2f} km/s")
```

```
Velocity dispersion ( ): 1202.58 km/s
```

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

[19]:
```
filtered_df['velocity_dispersion'].describe()
```

[19]:
```
count     137.000000
mean       -2.392825
std      1202.576295
min     -2803.471718
25%      -763.070775
50%       248.411265
75%       767.132350
max      4217.366753
Name: velocity_dispersion, dtype: float64
```

[20]:
```
print(f"The value of the cluster redshift = {z_cluster:.4f}")
```

```
The value of the cluster redshift = 0.0800
```

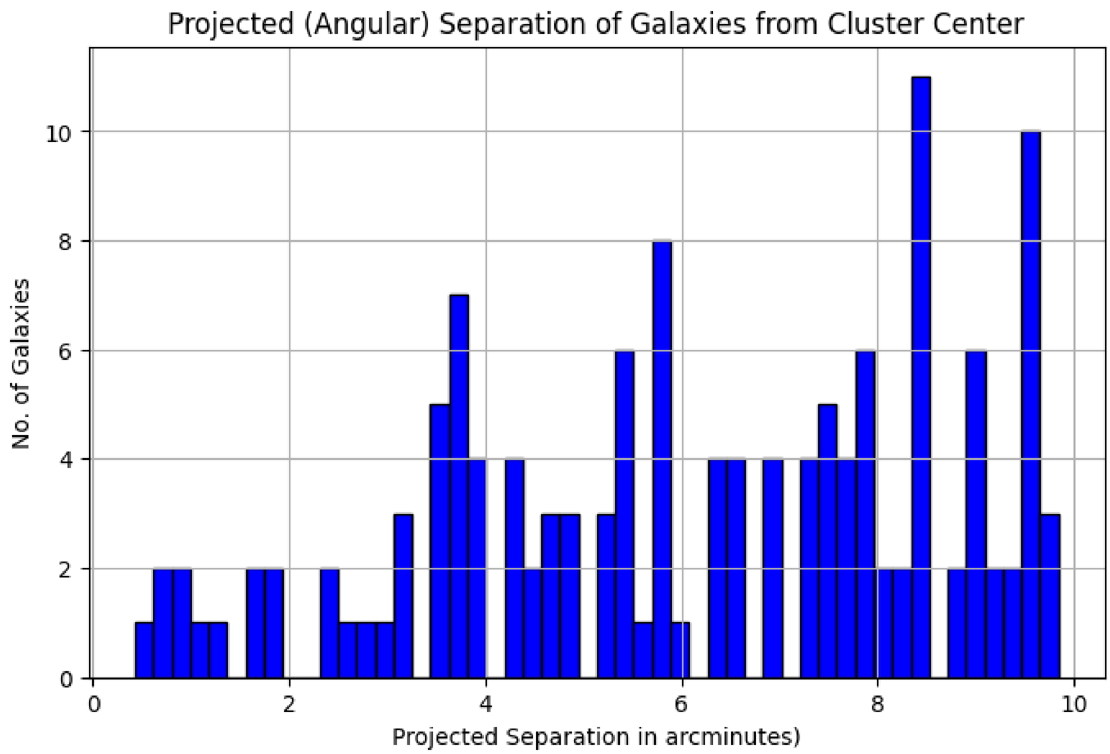
0.0.4 Step 4: Visualizing Angular Separation of Galaxies

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.

[21]:
```
#Plot histogram for proj sep column

# Histogram of projected angular separation
plt.figure(figsize=(8, 5))
plt.hist(filtered_df['proj_sep'], bins=50, color='blue', edgecolor='black')
```

```
plt.xlabel('Projected Separation in arcminutes)')
plt.ylabel('No. of Galaxies')
plt.title('Projected (Angular) Separation of Galaxies from Cluster Center')
plt.grid(True)
plt.show()
```



0.0.5 Determining size and mass of the cluster:

0.0.6 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 \$ \$ 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.

[22]:
```
r = (c_to_kms * z_cluster / H0) * (1 - (z_cluster / 2) * (1 + q0))#co-moving␣
↪distance
DA = r / (1 + z_cluster)  # angular diameter distance
theta_arcmin = filtered_df['proj_sep'].max()  # e.g., 9.8 arcmin
theta_rad = np.deg2rad(theta_arcmin / 60)     # converting to radians
diameter_mpc = DA * theta_rad #angular diameter (in arcminutes) into physical␣
↪diameter
print(f"Co-moving distance (r): {r:.2f}")
print(f"Angular diameter distance (D_A): {DA:.2f}")
print(f"Physical diameter of cluster: {diameter_mpc.value:.4f} Mpc (for   =␣
↪{theta_arcmin} arcmin)")
```

```
Co-moving distance (r): 336.35 Mpc
Angular diameter distance (D_A): 311.42 Mpc
Physical diameter of cluster: 0.8918 Mpc (for   = 9.844518658 arcmin)
```

0.0.7 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: - $ $ 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.

[26]:
```
# Constants
solar_mass = u.Msun      # 1 solar mass in kg
#Velocity dispersion ( ) in m/s
sigma = sigma_v * 1000* u.m / u.s  # converting to m/s
G_si = const.G
#Radius in meters (half of physical diameter)
radius_m = (diameter_mpc / 2)  # 1 Mpc = 3.086e22 meters
radius_m_in_meters = radius_m.to(u.m)
#Dynamical mass calculation
M_dyn_kg = (3 * sigma**2 * radius_m_in_meters) / G_si

#Dynamic mass calculation
M_dyn_solar = M_dyn_kg.to(solar_mass)
print(f"Cluster radius: {radius_m_in_meters:.2e}")
```

```python
print(f"Dynamical Mass: {M_dyn_solar:.2e}")
```

```
Cluster radius: 1.38e+22 m
Dynamical Mass: 4.50e+14 solMass
```

#LUMINOUS MASS CALCULATION

[27]:
```python
# Solar magnitude in r-band
M_sun_r = 4.65

#Computing luminosity distance in parsecs
DL_Mpc = (1 + filtered_df['specz']) * r  # r from earlier step
DL_pc = DL_Mpc * 1e6

#Computing absolute magnitude
filtered_df['M_r'] = filtered_df['rmag'] - 5 * np.log10(DL_pc) + 5

#Computing luminginosity in solar units
filtered_df['L_r'] = 10 ** (-0.4 * (filtered_df['M_r'] - M_sun_r))

#Estimation of luminous mass
#Typical M/L for elliptical galaxies  3 (can vary)
M_L_ratio = 3
filtered_df['luminous_mass'] = filtered_df['L_r'] * M_L_ratio

#Total luminous mass calculation
M_luminous_total = filtered_df['luminous_mass'].sum()
print(f"Total luminous mass: {M_luminous_total:.2e} solar masses")
```

```
Total luminous mass: 1.07e+13 solar masses
```

```
/tmp/ipython-input-27-553261645.py:9: 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_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df['M_r'] = filtered_df['rmag'] - 5 * np.log10(DL_pc) + 5
/tmp/ipython-input-27-553261645.py:12: 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_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df['L_r'] = 10 ** (-0.4 * (filtered_df['M_r'] - M_sun_r))
/tmp/ipython-input-27-553261645.py:17: 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_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_df['luminous_mass'] = filtered_df['L_r'] * M_L_ratio
```

Questions

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

H_0 = 72.97 ± 0.17 km/s/Mpc

2. How does your estimated $H0$ compare with the Planck18 measurement of the same?

The Planck 2018 measurement of the Hubble constant is **H_0 = 67.4 ± 0.5 km/s/Mpc**

Obtained H_0 value from the full dataset is **H_0 = 72.97 ± 0.17 km/s/Mpc**

The estimated value of H_0 is higher than the Planck18 measurement by **5.6 km/s/Mpc** and this shows a high tension between the local (supernova-based) and early-universe (CMB-based) measurements of the Hubble constant and this is known as the **Hubble tension.**

3. What is the age of the Universe based on your value of H_0? (Assume Ωm = 0.3). How does it change for different values of Ωm?

Age of Universe (with Ω_m = 0.3): 12.36 Gyr

Effect of Ω_m on age:

- If Ω_m increases universe will be younger
- If Ω_m decreases universe will be older

4. Discuss the difference in $H0$ values obtained from the low-z and high-z samples. What could this imply?

The Hubble constant from low-z supernovae is **H_0 = 73.01 km/s/Mpc.**

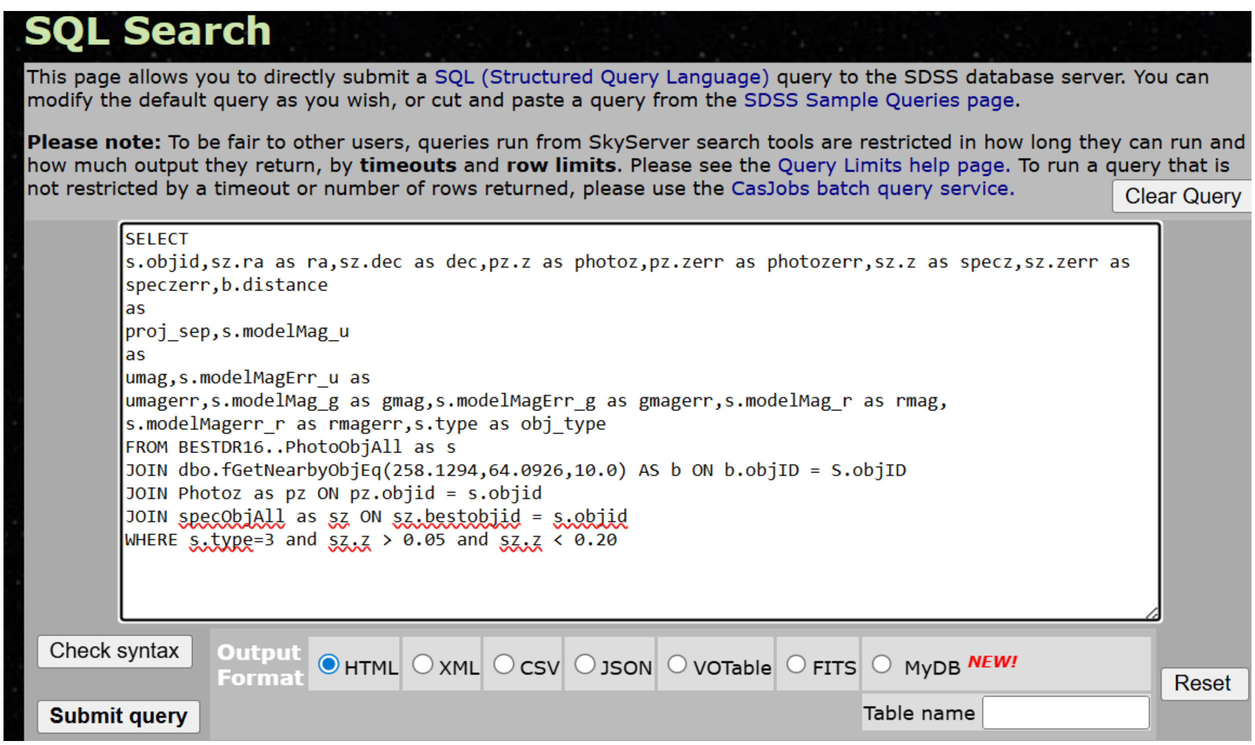
From high-z supernovae, it is **H_0 = 73.85 km/s/Mpc.**

Implications:

- Statistical fluctuations in the sample.
- Redshift-dependent systematics in supernova measurements.
- Possible evolution in supernova properties with redshift.
- Mild tension in cosmological model assumptions (e.g., flat ΛCDM).

INTERPRETATIONS AND OBSERVATIONS

Constants such as H0,c,G and q0 were first defined in the code to be used in the velocity dispersion calculation, estimating diameter of cluster and the final calculation of dynamical mass. According to [1] the best values of H0 lie in the range **73–75 km s−1 Mpc−1** . Hubble demonstrated the correlation between velocity displacement, also called redshift, and galaxy distance and this became known as Hubble constant. On the other hand, from detailed information available from the power spectrum of fluctuations in the cosmic microwave background, coupled with constraints favoring the existence of dark energy from distant supernova measurements, the precise prediction H0 is **67.4 ±1%** .The units of gravitational constant are deliberately converted to **Mpc·(km/s)^2·M_sun^{-1}** to calculate dynamic mass in solar mass. According to [2] The deceleration parameter, q0 quantifies how the expansion rate changes over time. After the definition of the constants the following sql query was pasted in SDSS SQL SERVER to obtain a csv file. This query selects a sample of galaxies within a specific redshift range around a particular point in the sky, for studies of galaxy properties . The redshift range (0.05-0.20) corresponds to cosmic distances of approximately 200-800 million light years.



In the csv file the column names didn’t start in the second row instead of the first row so the data had to be taken from the 2nd row.

#Table1														
objid	ra	dec	photoz	photozerr	specz	speczerr	proj_sep	umag	umagerr	gmag	gmagerr	rmag	rmagerr	obj_type
1.24E+18	257.8246	64.13326	0.079193	0.022867	0.082447	1.66E-05	8.347733	18.96488	0.043377	17.49815	0.005672	16.75003	0.004708	3
1.24E+18	257.8246	64.13326	0.079193	0.022867	0.082466	1.43E-05	8.347733	18.96488	0.043377	17.49815	0.005672	16.75003	0.004708	3
1.24E+18	257.8333	64.12604	0.091507	0.014511	0.081218	2.13E-05	8.011259	20.22848	0.072019	18.38334	0.007763	17.46793	0.005828	3
1.24E+18	257.8514	64.17325	0.081102	0.009898	0.079561	2.22E-05	8.739276	19.21829	0.050135	17.1897	0.004936	16.22043	0.003769	3
1.24E+18	257.8514	64.17325	0.081102	0.009898	0.079568	1.95E-05	8.739276	19.21829	0.050135	17.1897	0.004936	16.22043	0.003769	3

Using knowledge from [3] and [4] for each object id the mean of the spectroscopic **redshift/specz**, first value of **ra**, first value of **dec** and first value of **projected separation** between galaxies are aggregated. Using [5], the statistical summary of the aggregated mean value of **specz** is given and we can learn about the mean and standard deviation.

	objid	specz	ra	dec	proj_sep
0	1.240000e+18	0.081047	257.82458	64.133257	8.347733

	specz
count	1.000000
mean	0.081047
std	NaN
min	0.081047
25%	0.081047
50%	0.081047
75%	0.081047
max	0.081047

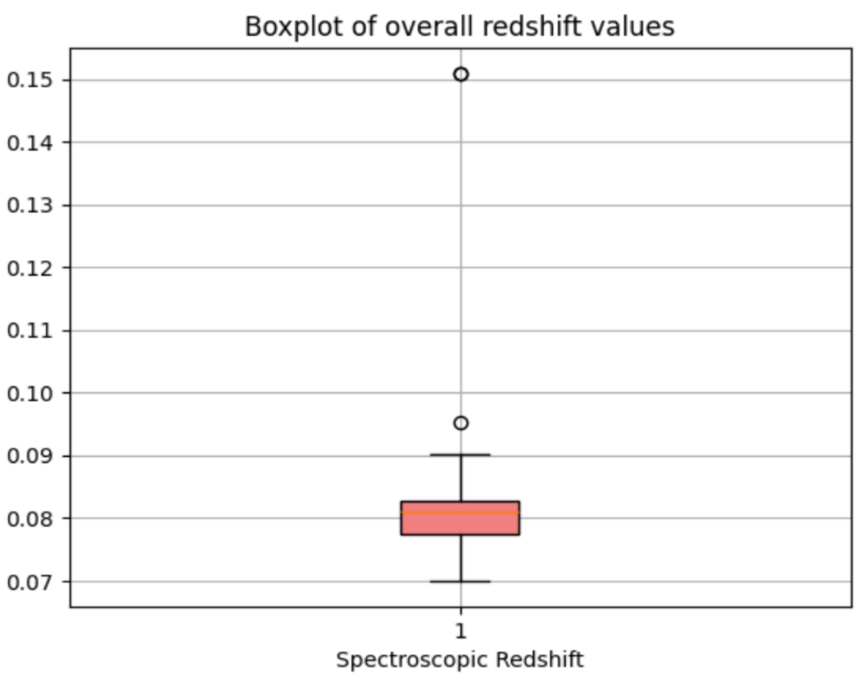
dtype: float64

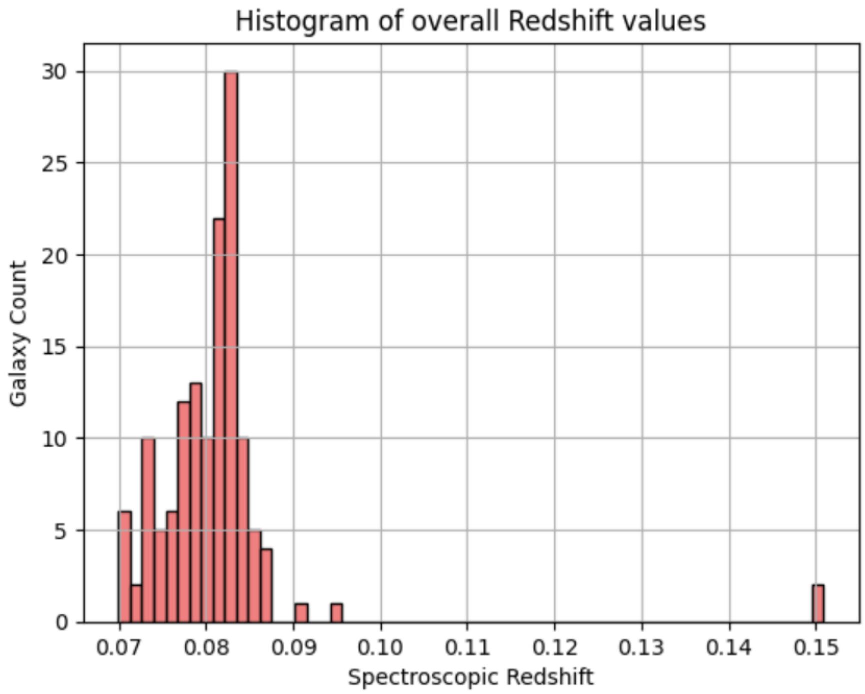
With the information in paper [6] to identify galaxies belonging to the same cluster, we assume that members will have similar redshifts, centered around a mean value, with small scatter. Astronomers often apply a **3-sigma clipping rule** :

- Any galaxy with a redshift within ±3 standard deviations (σ) from the mean (μ) is considered part of the cluster.
- Objects outside this range are likely foreground or background galaxies and are excluded.

```
Mean redshift: 0.08105
Standard deviation: 0.00950
3-sigma lower limit: 0.05255
3-sigma upper limit: 0.10954
```

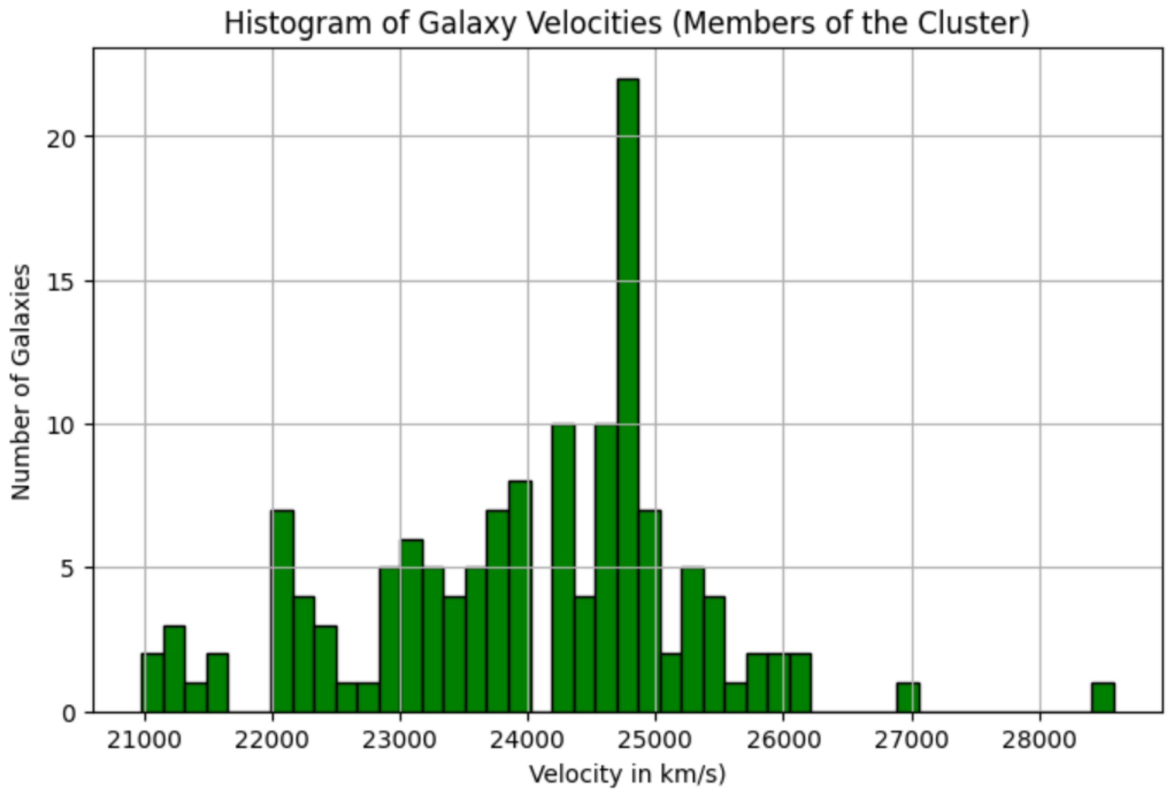
After this, a boxplot[7] is plotted to visually summarize the distribution of all the spectroscopic redshift (**specz**) values and a histogram[8] of redshift values is plotted to give a clear idea of how galaxies are distributed across different redshift ranges or in other words, how far away or how old the light from those galaxies is.





After this the data is filtered to identify cluster members based on their redshift values by applying a **3-sigma cut**. To this filtered data a new column called velocity is added which is calculated by multiplying the spectroscopic redshift value and the speed of light in km/s which had been defined prior. The new velocity column is then plotted using a histogram

	velocity
0	24717.081720
1	24722.783773
2	24348.666769
3	23851.808736
4	23853.802356



After this the values of velocity dispersion of galaxies in a cluster are calculated using the relativistic Doppler formula. Velocity dispersion is a measure of how fast and in how many different directions galaxies are moving relative to the system's average velocity[9].

For applying the relativistic Doppler formula, the average spectroscopic redshift of all galaxies belonging to a cluster(**z_cluster**) is computed and came out to be **0.800**

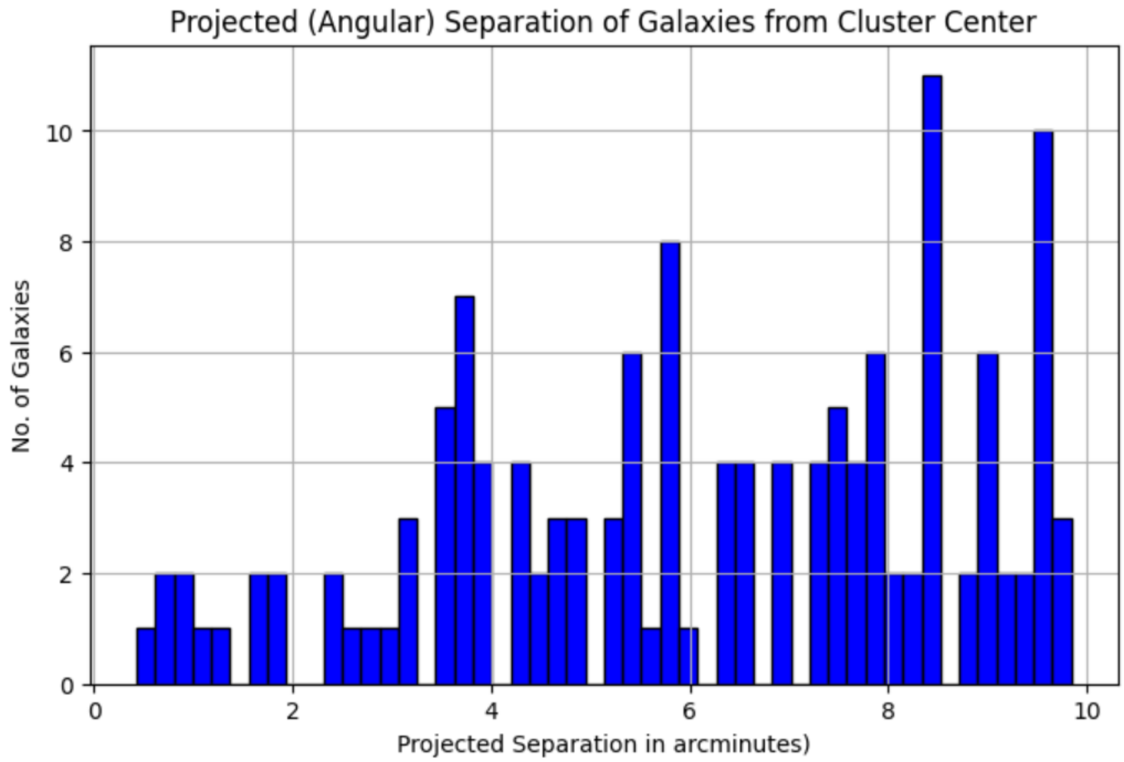
$$v = c \cdot \frac{(1+z)^2 - (1+z_{\text{cluster}})^2}{(1+z)^2 + (1+z_{\text{cluster}})^2}$$

A new column with the velocity dispersion values is added to the filtered data. The standard deviation of these values gives us a singular velocity dispersion value of **1202.58 km/s**. Using [5] the statistical summary of the aggregated mean value of velocity dispersion is given and we can learn about the mean and standard deviation.

	velocity_dispersion
count	137.000000
mean	-2.392825
std	1202.576295
min	-2803.471718
25%	-763.070775
50%	248.411265
75%	767.132350
max	4217.366753

dtype: float64

Next a histogram is plotted showcasing the projected/angular separation of galaxies from a cluster center. It helps us understand how galaxies are distributed around the cluster center. A peak near small separations means many galaxies are tightly clustered. Here, Projected separation is the minimum physical separation of two astronomical objects[11].



After this we try to estimate the physical diameter of the cluster using co moving distance(r),angular diameter distance(D_A) and angular size in radians(θ).

1. Co-moving Distance (r)

For low-redshift galaxies (where z<<1), we can approximate the **co-moving distance** using a Taylor expansion[12][13]:

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

2. Angular Diameter Distance (D_a)

This is the distance inferred from how large an object appears[12][13]:

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

It reflects how large the cluster appears to us given its distance and is crucial when translating angular sizes (in degrees or arcminutes) into physical sizes.

3. Converting Angular to Physical Diameter

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

This tells us the actual size of the cluster in space based on how wide it looks on the sky[13]. Theta (θ) represents the angular size of the galaxy cluster as seen from Earth. It tells us how large the cluster appears on the sky, usually in arcminutes or arcseconds. Theta (θ) is calculated by taking the maximum value of the projected separation values.

```
Co-moving distance (r): 336.35 Mpc
Angular diameter distance (D_A): 311.42 Mpc
Physical diameter of cluster: 0.8918 Mpc (for θ = 9.844518658 arcmin)
```

After this, to estimate the dynamical mass of a galaxy cluster, we apply the virial theorem[14], which relates the internal velocity dispersion of galaxies to the cluster's gravitational binding mass. For a spherically symmetric, virialized cluster, the mass can be approximated by:

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

The values of velocity dispersion, angular radius distance and Gravitational constant in SI units is used to calculate the dynamic mass. The result is converted to solar mass from kilograms. This approach assumes the cluster is in dynamical equilibrium and has an isotropic velocity distribution.

```
Cluster radius: 1.38e+22 m
Dynamical Mass: 4.50e+14 solMass
```

Finally, the total luminous mass of the galaxy cluster was estimated by first computing the absolute magnitudes of galaxies using their apparent r-band magnitudes and luminosity distances. These were then converted to luminosities in solar units using the solar absolute magnitude. Assuming a typical mass-to-light ratio of 3 for elliptical galaxies, each galaxy's luminous mass is calculated and summed up to get the total luminous mass of the cluster[15].

```
Total luminous mass: 1.07e+13 solar masses
```

REFERENCES

[1] arXiv:2305.11950 [astro-ph.CO]

[2] https://vickyscowcroft.github.io/PH40112_rmd/ch-obs-params.html

[3]https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html

[4] https://pandas.pydata.org/pandas-docs/version/0.22/generated/pandas.core.groupby.DataFrameGroupBy.agg.html

[5] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html

[6] https://iopscience.iop.org/article/10.1088/0004-637X/805/2/143

[7] https://www.geeksforgeeks.org/data-visualization/box-plot-in-python-using-matplotlib/

[8] https://www.geeksforgeeks.org/data-visualization/plotting-histogram-in-python-using-matplotlib/

[9] https://en.wikipedia.org/wiki/Velocity_dispersion

[10] https://ui.adsabs.harvard.edu/abs/1993ApJ...404...38G/abstract

[11] https://astronomy.stackexchange.com/questions/12122/what-is-projected-separation-and-how-can-i-make-sense-of-its-unit-h-1-kpc

[12] https://www.haus-der-astronomie.de/4171558/03

[13] arXiv:astro-ph/9905116

[14] https://www.aanda.org/articles/aa/full_html/2017/10/aa31104-17/aa31104-17.html

[15] arXiv:astro-ph/0606260

ASSIGNMENT 2

Predicting the Hubble Parameter and the Age of the Universe using Supernovae Ia Data

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:

```bash pip install numpy pandas matplotlib scipy astropy

##IMPORTING REQUIRED LIBRARIES

```
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 $\mu$, 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:

image.png

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

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

# Seeing the structure
file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1701 entries, 0 to 1700
Data columns (total 47 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   CID                  1701 non-null   object
 1   IDSURVEY             1701 non-null   int64
 2   zHD                  1701 non-null   float64
 3   zHDERR               1701 non-null   float64
 4   zCMB                 1701 non-null   float64
 5   zCMBERR              1701 non-null   float64
 6   zHEL                 1701 non-null   float64
 7   zHELERR              1701 non-null   float64
 8   m_b_corr             1701 non-null   float64
 9   m_b_corr_err_DIAG    1701 non-null   float64
 10  MU_SH0ES             1701 non-null   float64
 11  MU_SH0ES_ERR_DIAG    1701 non-null   float64
 12  CEPH_DIST            1701 non-null   float64
 13  IS_CALIBRATOR        1701 non-null   int64
 14  USED_IN_SH0ES_HF     1701 non-null   int64
 15  c                    1701 non-null   float64
 16  cERR                 1701 non-null   float64
 17  x1                   1701 non-null   float64
 18  x1ERR                1701 non-null   float64
 19  mB                   1701 non-null   float64
```

```
 20  mBERR                    1701 non-null   float64
 21  x0                       1701 non-null   float64
 22  x0ERR                    1701 non-null   float64
 23  COV_x1_c                 1701 non-null   float64
 24  COV_x1_x0                1701 non-null   float64
 25  COV_c_x0                 1701 non-null   float64
 26  RA                       1701 non-null   float64
 27  DEC                      1701 non-null   float64
 28  HOST_RA                  1701 non-null   int64
 29  HOST_DEC                 1701 non-null   int64
 30  HOST_ANGSEP              1701 non-null   float64
 31  VPEC                     1701 non-null   float64
 32  VPECERR                  1701 non-null   int64
 33  MWEBV                    1701 non-null   float64
 34  HOST_LOGMASS             1701 non-null   float64
 35  HOST_LOGMASS_ERR         1701 non-null   float64
 36  PKMJD                    1701 non-null   float64
 37  PKMJDERR                 1701 non-null   float64
 38  NDOF                     1701 non-null   int64
 39  FITCHI2                  1701 non-null   float64
 40  FITPROB                  1701 non-null   float64
 41  m_b_corr_err_RAW         1701 non-null   float64
 42  m_b_corr_err_VPEC        1701 non-null   float64
 43  biasCor_m_b              1701 non-null   float64
 44  biasCorErr_m_b           1701 non-null   float64
 45  biasCor_m_b_COVSCALE     1701 non-null   float64
 46  biasCor_m_b_COVADD       1701 non-null   float64
dtypes: float64(39), int64(7), object(1)
memory usage: 624.7+ KB

/tmp/ipython-input-4-2574597484.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
  file = pd.read_csv(file_path, delim_whitespace=True, comment="#")
```

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.

```python
for column in file.columns: #printing all columns in a neat way
    print(column)
```

```
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
```

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.

```
# Filter for entries with usable data based on the required columns

#Cleaning the columns
file_clean = file.dropna(subset=['zHD', 'MU_SH0ES',
'MU_SH0ES_ERR_DIAG']) #rows with NA/missing values are dropped using
dropna

#Extracting relevant data
Z = file_clean['zHD'].values
MU = file_clean['MU_SH0ES'].values
MU_ERR = file_clean['MU_SH0ES_ERR_DIAG'].values

print(MU) #checking the values of MU AND MU_ERR
print(MU_ERR)
```

```
[28.9987 29.0559 30.7233 ... 45.4865 45.4233 46.1828]
[1.51645  1.51747  0.782372 ... 0.281981 0.358642 0.281309]
```

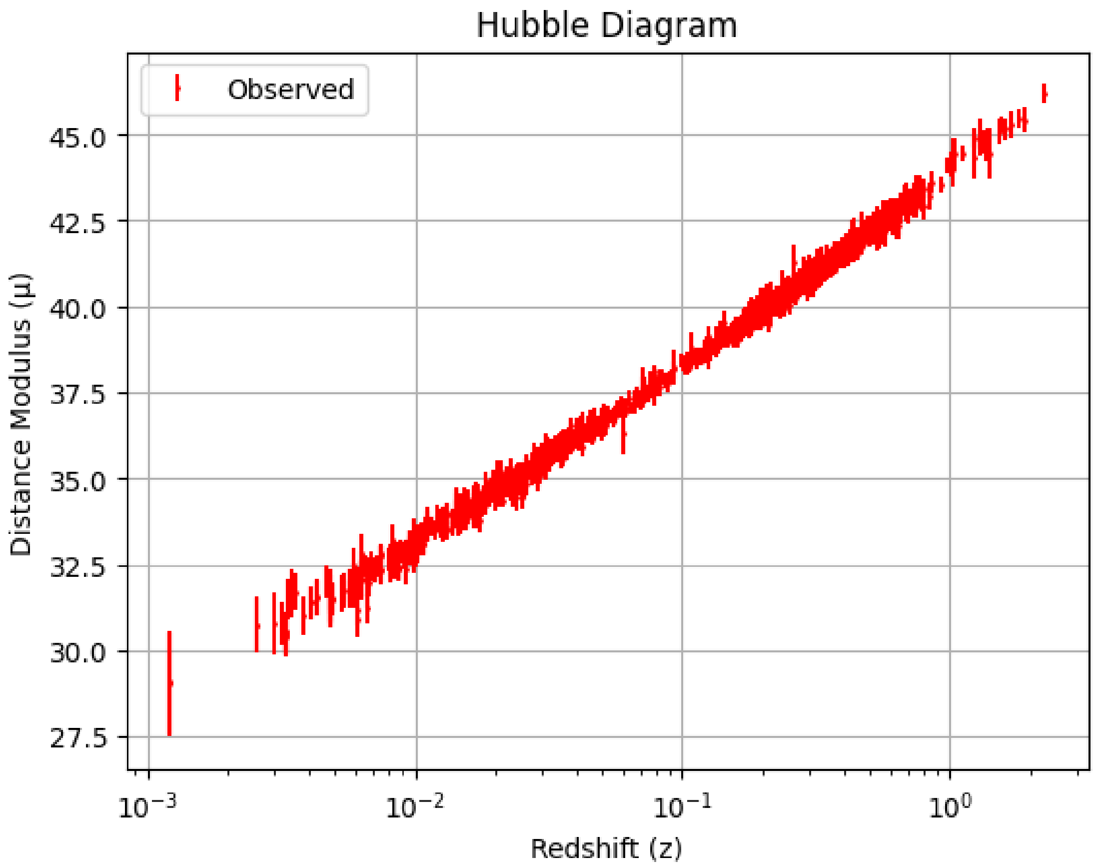
Plot the Hubble Diagram

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.

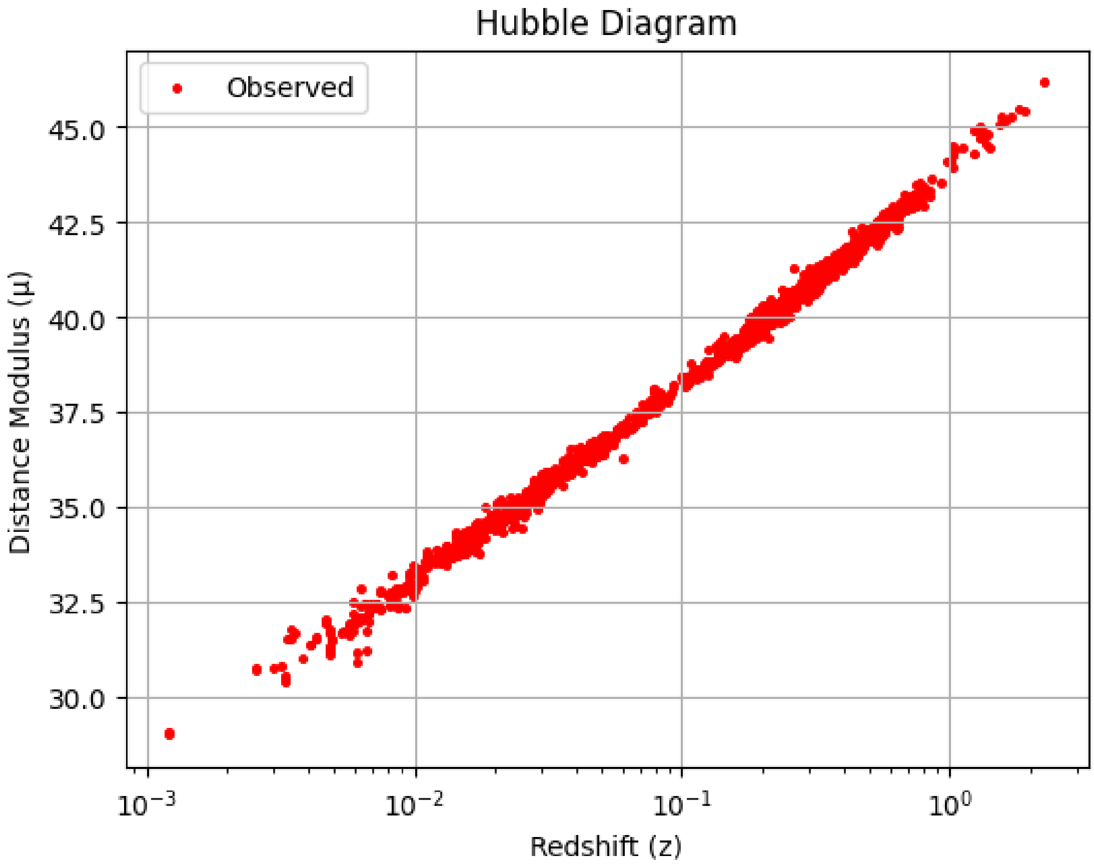
We use a logarithmic scale on the redshift axis to clearly display both nearby and distant supernovae.

```
#Errorbar plot
#errorbar is plotted to show the uncertainity in the data points


import matplotlib.pyplot as plt
plt.errorbar(Z, MU, yerr=MU_ERR, fmt='o',
markersize=1,color='red',label='Observed') #yerr is vertical bar to
check to show distance modulus uncertainity
#fmt and markersize are used to make the graph better. points in the
graph are marked with o of size=3
plt.xscale('log') #sets the scale of x-axis to log for better graph
visuals
plt.xlabel("Redshift (z)")
plt.ylabel("Distance Modulus (μ)")
plt.title("Hubble Diagram")
plt.legend() #helps show the label
```

```
plt.grid(True)
plt.show()
```




```
#Scatter plot without MU_ERR

import matplotlib.pyplot as plt
plt.scatter(Z, MU,color='red',s=7,label='Observed')
plt.xscale('log') #sets the scale of x-axis to log for better graph
visuals
plt.xlabel("Redshift (z)")
plt.ylabel("Distance Modulus (μ)")
plt.title("Hubble Diagram")
plt.grid(True)
plt.legend()
plt.show()
```



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:

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

- 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.

```
# Speed of light in km/s
c = 299792.458
from scipy.integrate import quad
# Define the E(z) for flat LCDM
# ΛCDM means lambda cold dark matter and is a math model of the big bang theory. Λ-cosmological constant due to dark energy. CDM-cold dark matter
def E(z, Omega_m):
  return np.sqrt(Omega_m * (1 + z)**3 + (1 - Omega_m))

# Luminosity distance in Mpc, scipy quad was used to integrate.
def luminosity_distance(z, H0, Omega_m):
  integral = np.array([quad(lambda z_: 1.0 / E(z_, Omega_m), 0, z_i)[0] for z_i in z])
  dL = (c / H0) * (1 + z) * 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)
  return 5 * np.log10(dL) + 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 , \text{km/s/Mpc} $
- $\Omega_m = 0.3$

```
from scipy.optimize import curve_fit

def fit_func(z, H0, Omega_m):
    return mu_theory(z, H0, Omega_m)

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

# Write a code for fitting and taking error out of the parameters
params, cov = curve_fit(fit_func, Z, MU, sigma=MU_ERR, p0=p0)#performing fitting
```

```python
#extracting the fitted values and uncertainities
H0_fit, Omega_m_fit = params
H0_err, Omega_m_err = np.sqrt(np.diag(cov))

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 ± 0.17 km/s/Mpc
Fitted Omega_m = 0.351 ± 0.012
```

Estimate the Age of the Universe

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

```python
from scipy.integrate import quad
import astropy.units as u
import numpy as np

def age_of_universe(H0, Omega_m):
    H0_SI = (H0 * u.km / u.s / u.Mpc).to(1 / u.s).value  # Converting
H0 from km/s/Mpc to 1/s

    #Integrating from z=0 to infinity
    integrand = lambda z: 1.0 / ((1 + z) * np.sqrt(Omega_m * (1 +
z)**3 + (1 - Omega_m)))
    integral, _ = quad(integrand, 0, np.inf)

    t0_seconds = integral / H0_SI #Age in seconds
    t0_gyr = (t0_seconds * u.s).to(u.Gyr).value  # Converting from
seconds to GYR

    return t0_gyr

# Calling the age_of_universe function with the best fit paramenters:
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
```

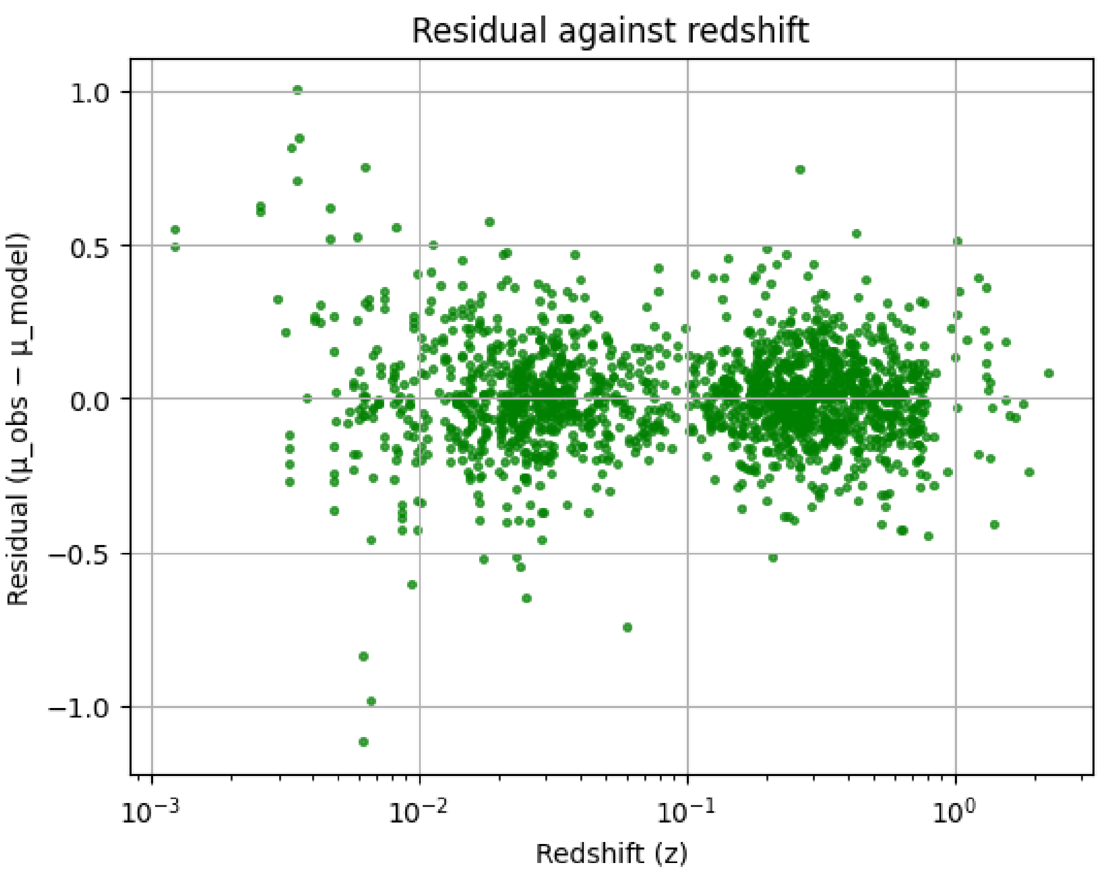
Analyze Residuals

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.

```
# Write the code to find residual by computing mu_theory and then plot
mu_model = fit_func(Z, H0_fit, Omega_m_fit)
residuals = MU - mu_model

#PLOT
plt.scatter(Z, residuals, s=7, color='green', alpha=0.7) #alpha is for
transparency
plt.xscale('log')
plt.xlabel('Redshift (z)')
plt.ylabel('Residual (μ_obs − μ_model)')
plt.title('Residual against redshift')
plt.grid(True)
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 $.

```
from scipy.optimize import curve_fit
import numpy as np

# Function with fixed Omega_m
def mu_fixed_Om(z, H0):
    return mu_theory(z, H0, Omega_m=0.3)

# Initial guess for H0
p0 = [70]

# Perform the fit (only H0 is varied)
params_fixed, cov_fixed = curve_fit(mu_fixed_Om, Z, MU, sigma=MU_ERR,
p0=p0, absolute_sigma=True)

# Extract fitted H0 and its uncertainty
H0_fixed = params_fixed[0]
H0_fixed_err = np.sqrt(cov_fixed[0, 0])

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

```
Fitted H0 with Ωm=0.3: 73.53 ± 0.17 km/s/Mpc
```

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.

```
from scipy.optimize import curve_fit

# Split point
z_split = 0.1

# Redshift and distance modulus data are extracted
Z = file_clean['zHD'].values
MU = file_clean['MU_SH0ES'].values
MU_ERR = file_clean['MU_SH0ES_ERR_DIAG'].values

# Low-z supernovae (z < 0.1)
```

```python
mask_low = Z < z_split
Z_low = Z[mask_low]
MU_low = MU[mask_low]
MU_ERR_low = MU_ERR[mask_low]

# High-z supernovae (z >= 0.1)
mask_high = Z >= z_split
Z_high = Z[mask_high]
MU_high = MU[mask_high]
MU_ERR_high = MU_ERR[mask_high]

# Fit only H0 with fixed Ωm = 0.3
p0 = [70]  # Initial guess for H0

# Performing curve fitting
H0_low, _ = curve_fit(mu_fixed_Om, Z_low, MU_low, sigma=MU_ERR_low,
p0=p0, absolute_sigma=True)
H0_high, _ = curve_fit(mu_fixed_Om, Z_high, MU_high,
sigma=MU_ERR_high, p0=p0, absolute_sigma=True)

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

```
Low-z (z < 0.1): H₀ = 73.01 km/s/Mpc
High-z (z ≥ 0.1): H₀ = 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

Questions

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

H_0 = 72.97 ± 0.17 km/s/Mpc

2. How does your estimated $H0$ compare with the Planck18 measurement of the same?

The Planck 2018 measurement of the Hubble constant is **H_0 = 67.4 ± 0.5 km/s/Mpc**

Obtained H_0 value from the full dataset is **H_0 = 72.97 ± 0.17 km/s/Mpc**

The estimated value of H_0 is higher than the Planck18 measurement by **5.6 km/s/Mpc** and this shows a high tension between the local (supernova-based) and early-universe (CMB-based) measurements of the Hubble constant and this is known as the **Hubble tension.**

3. What is the age of the Universe based on your value of H_0? (Assume Ωm = 0.3). How does it change for different values of Ωm?

Age of Universe (with Ω_m = 0.3): 12.36 Gyr

Effect of Ω_m on age:

- If Ω_m increases universe will be younger
- If Ω_m decreases universe will be older

4. Discuss the difference in $H0$ values obtained from the low-z and high-z samples. What could this imply?

The Hubble constant from low-z supernovae is **H_0 = 73.01 km/s/Mpc.**

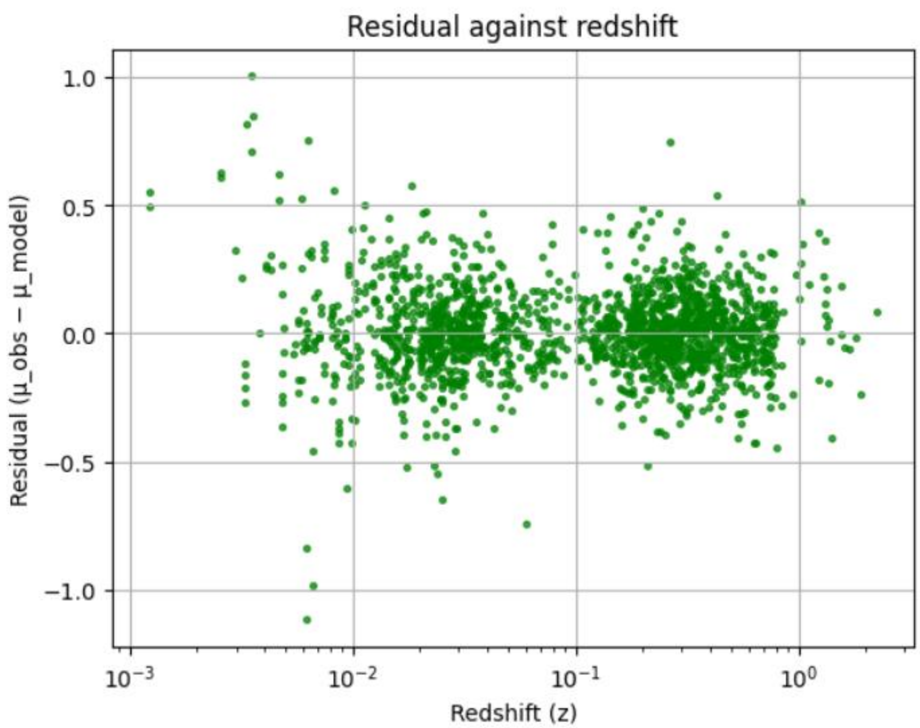
From high-z supernovae, it is **H_0 = 73.85 km/s/Mpc.**

Implications:

- Statistical fluctuations in the sample.
- Redshift-dependent systematics in supernova measurements.
- Possible evolution in supernova properties with redshift.
- Mild tension in cosmological model assumptions (e.g., flat ΛCDM).

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

```
# Write the code to find residual by computing mu_theory and then plot
mu_model = fit_func(Z, H0_fit, Omega_m_fit)
residuals = MU - mu_model

#PLOT
plt.scatter(Z, residuals, s=7, color='green', alpha=0.7) #alpha is for transparency
plt.xscale('log')
plt.xlabel('Redshift (z)')
plt.ylabel('Residual (μ_obs − μ_model)')
plt.title('Residual against redshift')
plt.grid(True)
plt.show()
```



Trends & Anomalies :

- Random scatter around zero meaning the model fits the data well.
- There's no clear upward or downward curve, supporting the adequacy of the ΛCDM model.
- Slightly increased spread at low and high redshifts – Residuals appear more scattered (more variance) at very low z ≲0.01 and very high z≳0.5z , indicating:
  - Higher observational uncertainties at extreme redshifts.
  - Possible redshift-dependent systematics.

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

- **Flat Universe :** Assuming flatness simplifies calculations. Allowing curvature could shift H_0 estimates.
- **Constant Dark Energy (Λ):**Assumes Λ doesn't evolve. A dynamic dark energy model could change μ(z) predictions.
- **Fixed Matter Density :** Used to reduce parameter degeneracy. Letting it vary introduces more uncertainty.
- **Uniform Supernova Properties:** Assumes no evolution with redshift. Evolution could bias distance estimates.

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

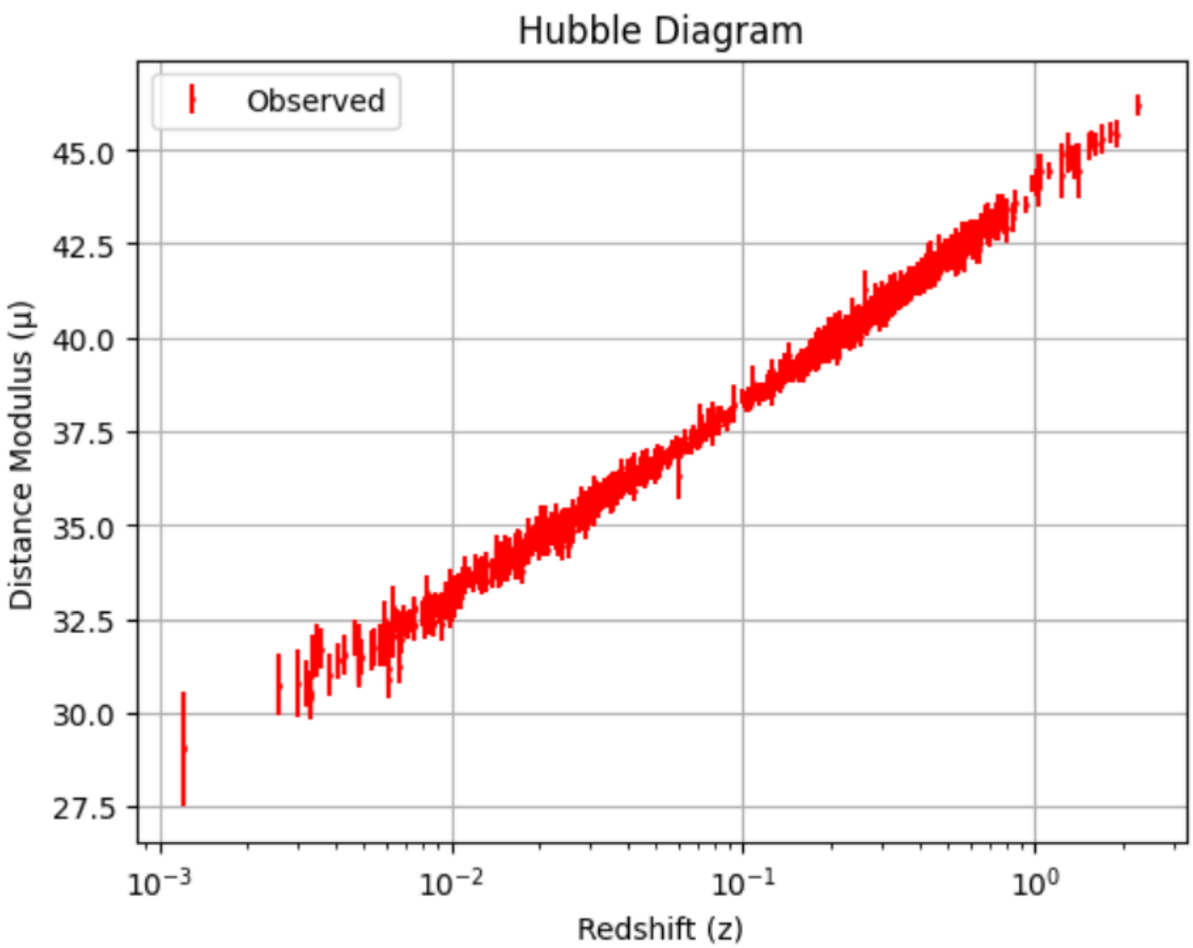
The redshift-distance relation shows that far away supernovae appear dimmer than expected in a uniformly expanding universe, implying the expansion of the Universe is accelerating which could be due to dark energy

INTERPRETATIONS AND OBSERVATIONS

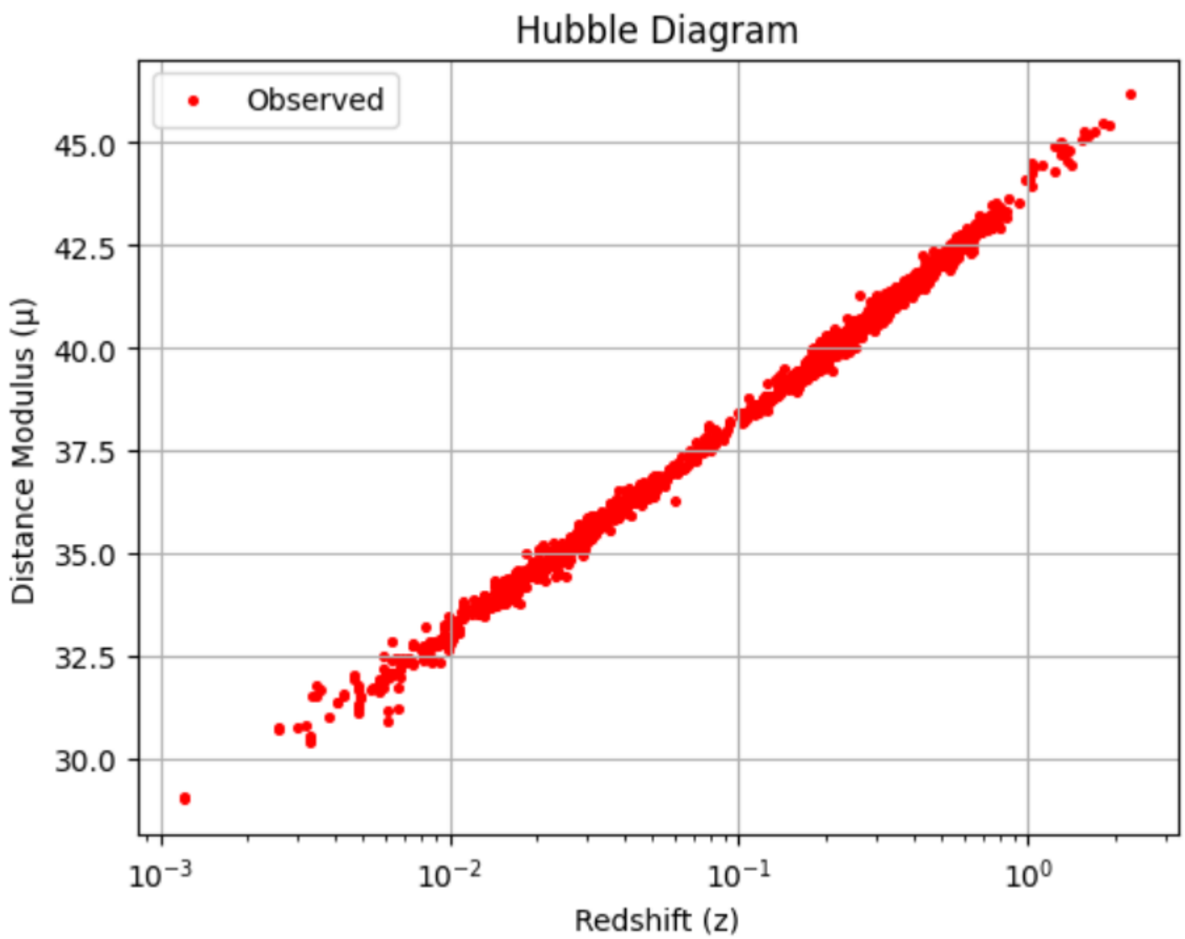
Necessary libraries like NumPy, Pandas, SciPy, AstroPy and matplotlib were installed and imported. SciPy is built to work with NumPy arrays and for fitting cosmological models and integrating equations [1]. The Astropy Project is a community effort to develop a single package for Astronomy in Python. It contains core functionality and common tools needed for performing astronomy and astrophysics research with Python[2]. It's used in this project for physical constants and unit conversions.

The **Pantheon+SH0ES.dat** dataset of **Type Ia supernovae** is used for measuring the cosmological parameters. The **Pantheon+SH0ES (Pantheon+)** dataset is an extensive compilation of **Type Ia supernovae (SNe Ia)** observations, serving as an important tool for probing the large-scale properties of the Universe. It contains 1701 light curves corresponding to 1550 distinct supernovae, covering a wide redshift range from 0.001 to 2.26. Compared to the original **Pantheon** dataset, **Pantheon+** significantly expands the number of low-redshift supernovae, particularly below redshift 0.08, by incorporating data from five major surveys, including the Foundation Supernova Survey, SOUSA, LOSS1, LOSS2, and the Dark Energy Survey (DES). This enrichment at low redshift makes it particularly valuable for investigating local cosmological features, such as the **Hubble constant (H_0)** and **matter density (Ω_m)**[3]. The dataset is read using read_csv[4] to read the data like a table and the columns are split using a whitespace.

The column names of the dataset are listed out in order to understand the data better. The key columns we used are **zHD, MU_SH0ES, MU_SH0ES_ERR_DIAG** which represents the redshift for the **hubble diagram**, distance modulus and uncertainty in the distance modulus respectively. Using **dropna()** method[5] the null values in rows of the key columns are removed/dropped. Following this the non-null data is extracted into new variables **Z,MU_ERR AND MU_ERR**. Next a Hubble diagram is plotted. The Hubble Diagram is a fundamental plot in observational cosmology that illustrates the relationship between **redshift (z)** and **distance modulus (μ)**[6], providing insight into the expansion history of the Universe. This plot helps visualize how far supernovae are relative to their redshifts and allows us to test different cosmological models against observations. Here, plt.errorbar[7][8][9] is used to plot the observed supernova data, with red dots representing each supernova and vertical error bars (**yerr=MU_ERR**) indicating the uncertainties in the distance modulus measurements. The redshift axis is set to a logarithmic scale (**plt.xscale('log')**) to clearly capture both nearby and distant supernovae on the same plot. Labels, a grid, and a legend are added for clarity. This visualization is crucial for comparing observational data with theoretical predictions of the Universe's expansion.



Hubble Diagram is also plotted using a scatter plot[10] to visualize the relationship between **redshift (z)** and distance **modulus (μ)** for observed supernovae. It plots each supernova as a red dot using the **scatter()** function, with the redshift values on the x-axis and the corresponding distance moduli on the y-axis. The x-axis is displayed on a logarithmic scale to clearly show both nearby and distant galaxies, which helps in identifying trends over a wide range of redshifts. Labels are added to both axes, along with a title for context. A legend is included to indicate that the data points represent observed values, and a grid is displayed to improve readability. This version of the **Hubble Diagram** excludes uncertainty bars, offering a cleaner view of the distribution of supernova data.



This next section defines the theoretical cosmological model based on the flat **ΛCDM (Lambda Cold Dark Matter)** framework, which is the standard model of cosmology[11]. The model assumes a flat universe dominated by cold dark matter and dark energy (represented by the cosmological constant, Λ). The key component of this model is the dimensionless **Hubble parameter E(z)**, which describes how the expansion rate of the universe changes with redshift zzz, depending on the **matter density parameter Ω_m**. The **luminosity distance $d_L(z)$**, which relates to how far away objects appear due to the universe's expansion, is calculated by integrating over redshift using **scipy.integrate.quad**. **The distance modulus μ(z)**, a measure used in supernova cosmology, is then computed from the luminosity distance using the standard formula **$\mu=5\log_{10}(d_L/Mpc)+25$**. The code defines three functions: **E(z, Omega_m)** computes the Hubble parameter, **luminosity_distance()** calculates **$d_L(z)$**, and **mu_theory()** returns the theoretical distance modulus for given values of redshift, **H_0**, and **Ω_m**, enabling comparison with observed supernova data.

- 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 :

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

Next, we focus on fitting the theoretical cosmological model to the observed supernova data in order to estimate key cosmological parameters—**the Hubble constant (H_0)** and **the matter density parameter (Ω_m)**. The fitting uses the **curve_fit** function from **scipy.optimize**[12], which performs **non-linear least squares optimization**. The model function **fit_func(z, H0, Omega_m)** returns the theoretical distance modulus **μ(z)** computed using the flat **ΛCDM** model. The observed data consists of **redshift values (Z)**, corresponding **distance moduli (MU)**, and **measurement uncertainties** (**MU_ERR**). An initial guess is provided for the parameters (**H_0 = 70 km/s/Mpc, Ω_m = 0.3**). After the fit, the best-fit values and their uncertainties are extracted by taking the square roots of the diagonal elements of the **covariance matrix (cov).** The output shows the fitted values for the **Hubble constant** and **matter density**, along with their respective uncertainties, providing insight into how well the model aligns with the observational data.

```
Fitted H0 = 72.97 ± 0.17 km/s/Mpc
Fitted Omega_m = 0.351 ± 0.012
```

Next we figure out how to estimate the **age of the Universe** based on the best-fit cosmological parameters **H_0 (Hubble constant)** and **Ω_m (matter density parameter)**, obtained from supernova data. According to the **ΛCDM** cosmological model, the age of the universe **t_0** can be computed by integrating the inverse of the

Hubble parameter over redshift, from the present (z = 0) to the very early universe (z → ∞). This is represented by the integral[13]:

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

First, H_0 is converted into SI units (1/s), since it is originally given in km/s/Mpc. Then the expression is numerically integrated for 1/((1+z)H(z)), where H(z) follows the **ΛCDM** relation[13]:

$$H(z) = H_0\sqrt{\Omega_m(1+z)^3 + (1-\Omega_m)}$$

This form assumes a flat universe with a cosmological constant (dark energy) and matter. The resulting time in seconds is then converted to **gigayears (Gyr)**. This gives a theoretical estimate for the age of the universe, which can be compared to other estimates such as those derived from **cosmic microwave background (CMB)** data by missions like **Planck**. Finally, the calculated age is mentioned[13].

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

Next we focus on **analyzing the residuals**—the difference between observed and model-predicted values of the **distance modulus (μ)**—to assess how well the **ΛCDM** cosmological model fits the supernova data[14][15]. The residual is computed as:

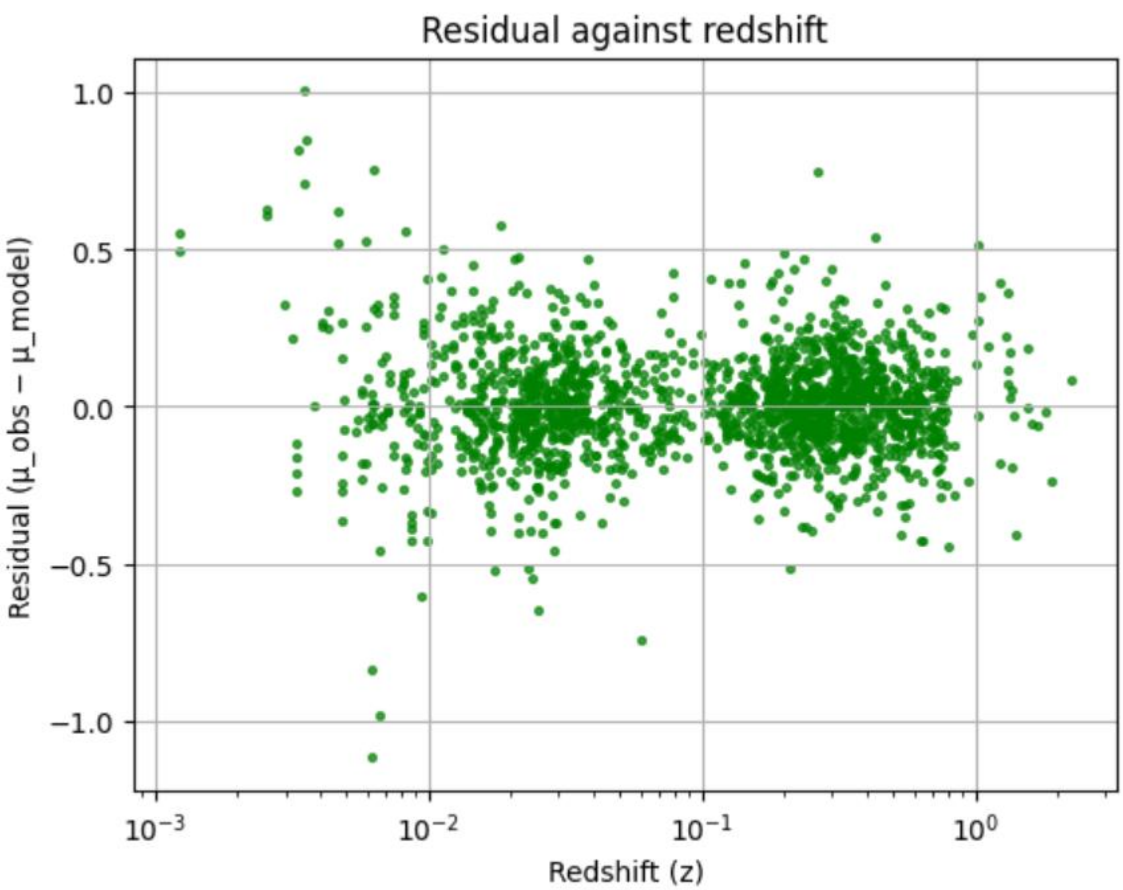
$$\text{Residual} = \mu_{\text{obs}} - \mu_{\text{model}}$$

where:

- $\boldsymbol{\mu_{obs}}$ is the observed distance modulus from supernova data.

- μ_{model} is the theoretical distance modulus calculated from the fitted **ΛCDM** model.

mu_model is computed using the best-fit values for $\mathbf{H_0}$ and $\mathbf{\Omega_m}$ from the previous fitting step. Then, the residuals are calculated by subtracting these model predictions from the observed values. A scatter plot is then generated with redshift on the x-axis (log scale) and the residuals on the y-axis. This plot helps visually identify any systematic deviations, biases, or clustering in the residuals. Ideally, for a good model fit, these points should scatter randomly around zero, indicating no systematic errors or trends. If noticeable patterns appear, it might suggest that the model needs refinement or that there are issues in the data.



In the next step, the aim was to estimate the **Hubble constant ($\mathbf{H_0}$)** more precisely by **fixing the matter density parameter ($\mathbf{\Omega_m}$) to a commonly accepted value of 0.3**. This approach helps **reduce degeneracy between parameters**—a situation where multiple parameter combinations can produce similar fits. By fixing $\mathbf{\Omega_m}$, we isolate the effect of H_0 on the distance modulus–redshift relation, improving confidence in its estimation.

A function **mu_fixed_Om** is defined that returns the theoretical **distance modulus μ(z)** based on the **ΛCDM** model, using a fixed $\boldsymbol{\Omega_m = 0.3}$ and variable H_0[16]. The **curve_fit** function from **scipy.optimize** is then used to perform **non-linear least squares fitting** of this model to the supernova data **(Z and MU)**, taking into account observational uncertainties **(MU_ERR)**. Only H_0 is fitted, starting with an initial guess of 70. Finally, the fitted value of H_0 and its uncertainty are printed. This approach provides a cleaner constraint on the Hubble constant assuming a standard cosmological matter density.

```
Fitted H0 with Ωm=0.3: 73.53 ± 0.17 km/s/Mpc
```

The final step investigates whether the value of the Hubble constant H_0 differs when measured from low-redshift versus high-redshift **Type Ia supernovae**, which can reveal possible redshift-dependent tension in cosmological observations. The dataset is split into two subsets: one containing **low-redshift supernovae** with **$z<0.1$** and another with **high-redshift supernovae $z \geq 0.1$**[17]. For each subset, the **distance modulus μ** is fitted as a function of redshift using the flat **ΛCDM** cosmological model, while **fixing the matter density parameter Ω_m to 0.3**, to reduce degeneracy. Using **curve_fit**, the script independently estimates H_0 for each redshift range. By comparing the resulting values of H_0, one can examine whether the Universe's expansion rate appears to evolve with redshift or if different parts of the cosmic history yield consistent estimates of the Hubble constant. This is a crucial analysis in the context of the current **Hubble tension**—the discrepancy between early- and late-universe measurements of H_0.

```
Low-z (z < 0.1): H₀ = 73.01 km/s/Mpc
High-z (z ≥ 0.1): H₀ = 73.85 km/s/Mpc
```

REFERENCES

[1] https://pypi.org/project/scipy/

[2] https://anaconda.org/anaconda/astropy

[3] arXiv:2310.11727

[4] https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

[5] https://www.aporia.com/resources/how-to/drop-rows-pandas-dataframe-column-vamue-nan/

[6] https://www.pnas.org/doi/epdf/10.1073/pnas.2536799100

[7] https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.errorbar.html

[8] https://www.w3schools.com/python/matplotlib_markers.asp

[9] https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xscale.html

[10] https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html

[11] https://academic.oup.com/mnras/article/537/1/436/7951521?login=false

[12]https://www.researchgate.net/publication/277722502_Marginal_evidence_for_cosmic_acceleration_from_Type_Ia_supernovae

[13] https://physics.stackexchange.com/questions/125970/estimating-age-of-the-universe-by-hubbles-law

[14] arXiv:1401.4064

[15] https://www.aanda.org/articles/aa/full_html/2021/05/aa38447-20/aa38447-20.html

[16] https://www.aanda.org/articles/aa/full_html/2024/01/aa47121-23/aa47121-23.html

[17]https://arxiv.org/pdf/2302.05709